UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de máster

Tecnologías IoT aplicadas a la gestión del tiempo para personas con TEA

Víctor Manuel Vázquez González

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

Universidad Autónoma de Madrid

Escuela Politécnica Superior

Proyecto para la obtención del título de
Máster en Ingeniería Informática
por la Universidad Autónoma de Madrid

Tutor del trabajo fin de máster:
Javier Gómez Escribano

Tecnologías IoT aplicadas a la gestión del tiempo para personas con TEA

Víctor Manuel Vázquez González

Víctor Manuel Vázquez González
Tecnologías IoT aplicadas a la gestión del tiempo para personas con TEA

Víctor Manuel Vázquez González
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mis padres y a mi hermana, por estar siempre ahí.

Agradecimientos

Antes de nada me gustaría hacer mención a todas aquellas personas que han hecho este trabajo de fin de máster posible, que han aportado su granito de arena, y me han ayudado en esta pequeña etapa del Máster de Ingeniería Informática.

Agradecer a mis familiares y amigos, por apoyarme en todo momento en la consecución de este gran objetivo y un gran paso en mi vida. En especial a mi mejor amiga Juli, por su apoyo constante, por sacarme sonrisas cuando más hacían falta, y por estar siempre ahí, por todo esto y mucho más.

También me gustaría destacar el trabajo de mi tutor Javier Gómez Escribano, ya que con su entrega, sus ideas, y sus ganas de ayudar, han hecho que la realización de este proyecto haya sido posible y el camino hasta la cima de la montaña haya sido más llevadero.

Por último quiero agradecer al departamento de Ingeniería Informática de la Universidad Autónoma de Madrid por la ayuda para la adquisición del material necesario para la realización de este Trabajo de Fin de Máster.

Gracias a todos.

Resumen

Este Trabajo de Fin de Máster se centra en la implementación de nuevas tecnologías de apoyo, exponiendo la posibilidad de la utilización de tecnologías y dispositivos electrónicos para la asistencia a personas con Trastorno del Espectro Autista (TEA). Este proyecto se orienta a mejorar la vida de estas personas, ayudándolas y asistiéndolas en tareas cotidianas que llegaban a ser complejas para ellos.

El principal objetivo de este proyecto es el de brindar un dispositivo de apoyo que ayude a colectivos con dificultades a mejorar la percepción del tiempo en la realización de actividades. Para conseguirlo se hace uso de pictogramas, imágenes que representan a las actividades que deben realizar y de las cuales ya se conoce su significado asociado. Para complementar el uso de los pictogramas, se ha desarrollado un sistema visual que además se centra en comunicar el tiempo restante que le queda a cada una de las actividades, ayudando en gran medida a establecer rutinas de actividades y a mejorar la percepción y gestión del tiempo para cada una de ellas, ya que se permite saber en todo momento cuando van a finalizar.

Este sistema está también orientado a facilitar la tarea de educadores y padres, ofreciendo tecnología de bajo coste, sencilla de utilizar, y que permite monitorizar en todo momento las actividades que se realizan. Para permitir el control del sistema mientras está en funcionamiento, se ha separado el visor de pictogramas de la aplicación de gestión, siendo esta última accesible por parte del usuario en todo momento y ofreciendo un control absoluto de la reproducción en tiempo real, lo que permite ayudar a gestionar el tiempo si fuera necesario. Para adaptarse a las necesidades de todas las personas, se ha desarrollado un sistema altamente personalizable, lo que da la posibilidad al usuario encargado de gestionarlo, buscar las configuraciones que mejor se adaptan a cada determinado grupo de personas que hará uso del sistema.

Palabras clave

Tecnologías de Apoyo, Pictogramas, Internet de las Cosas, Trastorno del Espectro Autista, Hazlo tú mismo, Raspberry Pi

Índice

Listas

Lista de códigos

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

1.1. Motivación

La realización de este proyecto viene motivada por las ganas de poder volcar mis conocimientos en el campo de la informática, en intentar ayudar a personas a realizar tareas que antes eran más complicadas para ellos.

Se espera que este Trabajo de Fin de Máster pueda aportar un nuevo dispositivo a lo que se conoce como las tecnologías de apoyo, las cuales van obteniendo cada vez más importancia debido a sus mejoras constantes, que ayude y facilite a que personas con discapacidad puedan realizar nuevas actividades que antes no podían, y les ayude a mejorar la percepción del tiempo.

1.2. Objetivos

El objetivo principal de este proyecto es el de implementar un sistema capaz de asistir a la realización de actividades de las personas con el Trastorno del Espectro Autista (TEA), tratando de que la utilización continuada del sistema les ayude a conseguir organizar y percibir mejor el tiempo que le dedican a las actividades. Para la consecución de este objetivo será necesario establecer objetivos mas específicos, cuyo cumplimiento llevará a alcanzar un mejor resultado final.

Estos objetivos específicos que llevan a la consecución del objetivo principal son:

- Análisis en profundidad de las necesidades específicas de las personas con TEA.
- Diseño e implementación de una aplicación de gestión y un visor de pictogramas.
- Montaje de un prototipo hardware que permita estudiar la interacción y viabilidad del sistema.
- Realización de un proyecto que se apoye en open-hardware de bajo coste que facilite las prácticas de Hazlo tú Mismo (Do It Yourself DIY).

Para cumplir con los objetivos propuestos será necesario realizar avances sobre las dos partes principales del sistema, las cuales se detallan a continuación.

En primer lugar será necesario el desarrollo de una parte **software**, la cual está compuesta por una aplicación principal de control y un servidor API, encargado de la comunicación con el sistema de almacenamiento de datos y control de las luces LED. Esta parte software debe permitir al usuario controlar la reproducción de pictogramas, gestionar listas de pictogramas que podrán ser reutilizadas, y por último, un apartado que facilite la personalización de las luces LED, ya que como se ha mencionado anteriormente, se deben de poder adaptar al uso de distintos tipos de usuario.

La última parte es la del **hardware**, donde se implementará un prototipo de dispositivo capaz de mostrar los pictogramas que reproduce el usuario junto a los componentes que permitan representar el tiempo de cada pictograma. Para el desarrollo de este prototipo se priorizará la utilización de hardware de bajo coste, por lo que se realizará un análisis de las mejores opciones del mercado y se valorará muy positivamente la flexibilidad en la posibilidad de hacer uso de distintos tipos de dispositivos.

1.3. Organización del documento

El presente documento está dividido en diez capítulos.

En el **primer** capítulo se puede encontrar una pequeña introducción al proyecto junto a las motivaciones y objetivos.

En el **segundo** capítulo, el cual contiene dos secciones, se realiza una planificación temporal del proyecto, así como un estudio de estimación de costes.

En el **tercer** capítulo se realiza un estudio sobre el estado del arte del tema, realizando un análisis sobre los proyectos existentes relacionados y las investigaciones publicadas.

En el **cuarto** capítulo se encuentra detallada la fase de especificación y análisis de requisitos, en donde se pueden encontrar detallados los requisitos que debe cumplir el sistema.

En el **quinto** capítulo se comienza con la realización de la fase de diseño del sistema desglosado en distintas secciones.

En el **sexto** capítulo se aborda la fase de implementación del sistema desglosado por las distintas partes que lo componen.

En el **séptimo** capítulo se detallan las pruebas de caja negra y caja blanca realizadas sobre el sistema desarrollado.

En el **octavo** capítulo se realizan las conclusiones finales del proyecto, junto a las vías de trabajo futuro.

2

Planificación y Costos

En este capítulo se realizará una estimación de los costes que supone la realización del proyecto, así como la planificación temporal del mismo separada por tareas y subtareas.

2.1. Planificación y seguimiento

Para la realización de este proyecto se va a seguir una metodología de desarrollo en cascada organizada en 6 fases distintas. Estas fases son análisis, diseño, desarrollo software, desarrollo hardware, pruebas y montaje.

La decisión de utilizar una metodología de desarrollo en cascada ha sido tomada teniendo en cuenta los siguientes factores:

- Es un modelo fácil de seguir y de comprender.
- Los requisitos del proyecto son estables, por lo que no se van a producir grandes cambios en medio del desarrollo.
- Permite estimar los costes al principio del proyecto de manera fácil y sencilla.
- Se cuenta con el conocimiento necesario para llevarlo a cabo

Una vez planificadas y conocidas las fases que seguirá el proyecto, el Diagrama de Gantt resultante es el que se puede apreciar en la Figura 2.1.

El Diagrama de Gantt permite observar la planificación inicial por semanas a lo largo del tiempo, lo que permite obtener una buena imagen general, por lo que para ser más preciso se mostrará la planificación por horas de cada una de las tareas y subtareas en la Tabla 2.1.

Tanto el diagrama de Gantt como la planificación de horas que se muestran en las figuras anteriores, muestran el tiempo estimado inicialmente, por lo que la desviación entre la planificación inicial y la final es la que se puede ver en la Tabla 2.2. Como se puede observar, a pesar de haber algunos errores en la estimación, estos no son demasiado elevados y en conjunto el error ha sido bastante bajo.

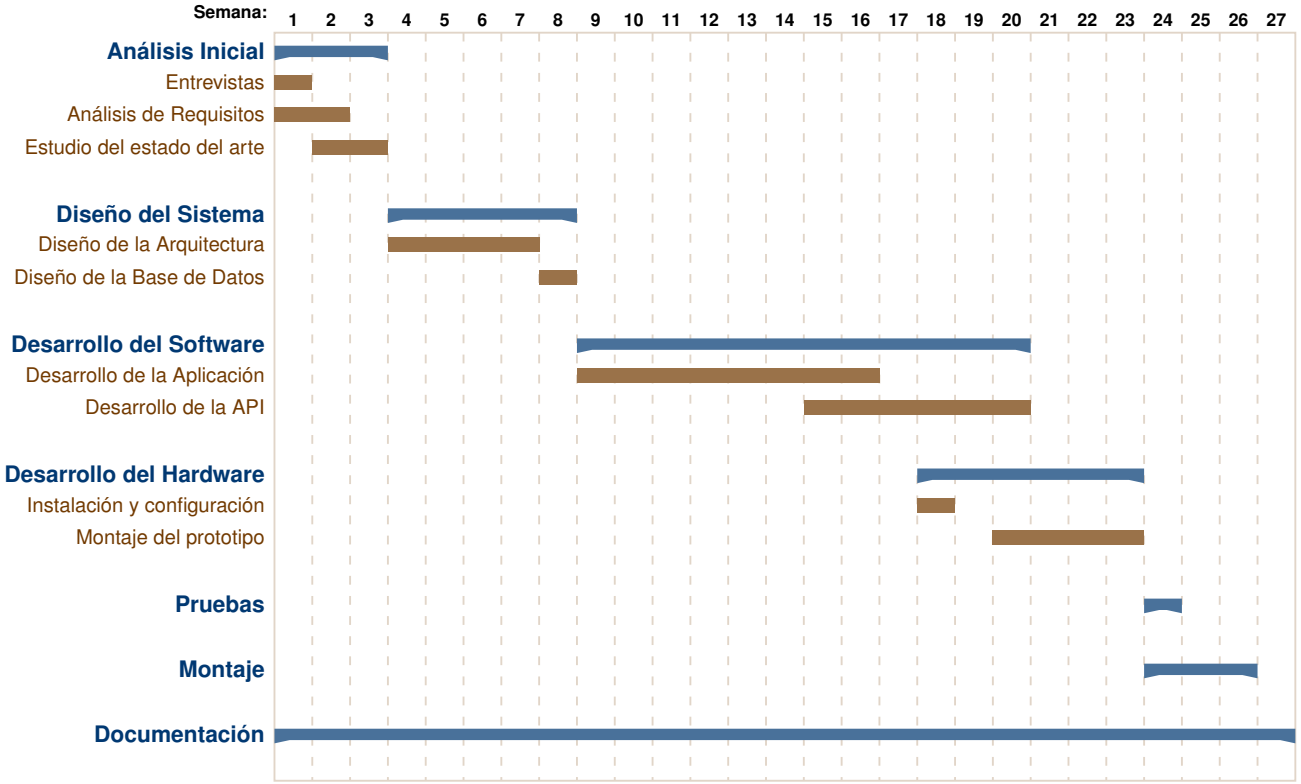

Figura 2.1: Diagrama de Gantt del Proyecto

Tareas/Subtareas	Horas
T1 Análisis inicial del problema	**33**
T1.1 Entrevistas	2
T1.2 Análisis de requisitos	6
T1.3 Estudio del estado del arte	25
T2 Diseño del sistema	**40**
T2.1 Diseño de la arquitectura	30
T2.2 Diseño de la base de datos	10
T3 Desarrollo del software	**120**
T3.1 Desarrollo de la aplicación	70
T3.2 Desarrollo del servidor	50
T4 Desarrollo del hardware	**40**
T4.1 Instalación y configuración del servidor	10
T4.2 Montaje del prototipo	30
T5 Realización de pruebas	**5**
T6 Implementación del proyecto	**7**
T7 Redacción de la memoria	**55**
TOTAL	**300**

Tabla 2.1: Tabla con la estimación inicial de horas para cada tares y subtarea

Tareas/Subtareas	Tiempo estimado	Tiempo empleado	Diferencia
T1 Análisis inicial del problema	**33**	**35**	**+2**
T1.1 Entrevistas	2	2	0
T1.2 Análisis de requisitos	6	7	+1
T1.3 Estudio del estado del arte	25	26	+1
T2 Diseño del sistema	**40**	**37**	**-3**
T2.1 Diseño de la arquitectura	30	29	-1
T2.2 Diseño de la base de datos	10	8	-2
T3 Desarrollo del software	**120**	**118**	**-2**
T3.1 Desarrollo de la aplicación	70	73	+3
T3.2 Desarrollo del servidor	50	45	-5
T4 Desarrollo del hardware	**40**	**45**	**+5**
T4.1 Instalación y configuración del servidor	10	12	+2
T4.2 Montaje del prototipo	30	33	+3
T5 Realización de pruebas	**5**	**8**	**+3**
T6 Implementación del proyecto	**7**	**6**	**-1**
T7 Redacción de la memoria	**55**	**50**	**-5**
TOTAL	**300**	**299**	**-1**

Tabla 2.2: Tabla de hora con la desviación entre la planificación inicial y el tiempo empleado finalmente

2.2. Estimación de costos

Para la realización de la estimación de costes del proyecto hay que tener en cuenta que existen dos aspectos que le afectan. El primer aspecto tiene que ver con el costo de tiempo que se le va a dedicar al proyecto, mientras que por el otro lado está el coste relativo al precio del hardware y software que se van a utilizar en el desarrollo del proyecto.

Para la realización de esta estimación se utilizará el método de estimación PERT (Program Evaluation Review Technique) [1], el cual utiliza tres valores de estimación de tiempo (pesimista, más probable y optimista) para sacar una estimación de tiempo de realización para cada tarea.

Para calcular la duración de la actividad, la estimación PERT utiliza la formula que se puede ver en la Figura 2.1, siendo tO la estimación optimista, tM la más probable y tP la estimación de tiempo más pesimista.

$$DuracionActividad = \frac{tO + 4tM + tP}{6} \tag{2.1}$$

Por tanto, teniendo en cuenta solo la parte de implementación del software y hardware, la estimación PERT quedaría de la siguiente manera:

En primer lugar, la estimación de la implementación del software se realizaría teniendo en cuenta los siguientes valores.

- Tiempo optimista: 90 horas
- Tiempo más probable: 120 horas
- Tiempo pesimista: 140 horas

$$DuracionActividad = \frac{90 + 4 * 120 + 140}{6} = 118{,}33 horas \tag{2.2}$$

Por tanto la estimación del PERT nos da una duración de actividad de 118 horas, por lo que a una jornada laboral de 8 horas diarias nos daría un desarrollo de 15 días de trabajo completos para el apartado software. Teniendo en cuenta que el salario del programador es de 28,00€/hora, el coste de desarrollo software del proyecto es el siguiente:

$$Coste() = 118 * 28{,}00 = 3304{,}00 \tag{2.3}$$

En cuanto al coste de la implementación del hardware, la estimación del PERT se calcularía de la

siguiente forma teniendo en cuenta estos nuevos valores del PERT:

- Tiempo optimista: 33 horas
- Tiempo más probable: 40 horas
- Tiempo pesimista: 58 horas

$$DuracionActividad = \frac{33 + 4 * 40 + 58}{6} = 41{,}83horas \tag{2.4}$$

Para este caso, la duración de la actividad que da la estimación PERT es de 41.83 horas, lo que son 6 jornadas de trabajo completas de 8 horas de duración. Teniendo en cuenta que el salario de la persona que ensambla y desarrolla el hardware también es de 28.00€/hora el coste del desarrollo hardware es el siguiente:

$$Coste() = 41{,}83 * 28{,}00 = 1171{,}24 \tag{2.5}$$

Por tanto, el coste total de la implementación sumando el desarrollo del software y el desarrollo del hardware resultante sería el siguiente:

$$CosteImplementacin = 1171{,}24 + 3304{,}00 = 4475{,}24 \tag{2.6}$$

Para el cálculo de los costes del hardware se utilizarán como referencia los precios que aparecen en la Tabla B.1 publicada en el apartado B. Teniendo en cuenta esta información, los dispositivos utilizados para el proyecto suponen un gasto total de **197.48 €**

En cuanto al software utilizado para la implementación del proyecto este tiene un coste gratuito, ya que se han utilizado herramientas gratuitas de código libre para toda la implementación.

3

Estado del Arte

Antes de comenzar con la Fase de Análisis se pondrá en contexto el marco del proyecto, haciendo un pequeño estudio sobre el Trastorno del Espectro Autista, y analizando las soluciones y tecnologías existentes que estén relacionadas con las de este proyecto.

3.1. Trastorno del Espectro Autista

El Trastorno del Espectro Autista es un trastorno del desarrollo, que según estiman ciertos estudios, afecta a uno de cada 160 niños y provoca deficiencias en la comunicación e interacción social, problemas en la flexibilidad del comportamiento, y comportamientos repetitivos. Para estas personas la realización de tareas cotidianas puede llegar a ser mucho más complicado que para otras personas, ya que su percepción de las cosas y del entorno puede ser diferente al del resto [2] [3].

Uno de los problemas comunes en las personas con TEA es la dificultad para percibir el tiempo, ya que el uso de relojes tradicionales es muy complejo para ellos y no asocian la posición de las flechas o los números con una determinada hora. Para ayudarles con este problema se han ideado distintas soluciones entre las que destaca el uso de códigos de colores para mejorar esta percepción temporal, ya que para estas personas es mucho más sencillo asociar colores para entender el tiempo restante de las tareas. Este tipo de códigos de colores pueden ser utilizados junto a relojes y cronómetros para ayudarles a establecer relaciones entre la percepción de colores, mucho más sencilla, y la hora o el tiempo que resta de actividad [4].

Debido a esto, se han ido realizando estudios que buscan maneras de mejorar la vida de estas personas mediante el uso de diferentes técnicas, como por ejemplo mediante el seguimiento de rutinas o, aprovechando cada vez más los nuevos avances tecnológicos, intentando implementar una mayor integración de la tecnología asistiva [5] como puede ser con la utilización de dispositivos táctiles para facilitar la comunicación [6]. Estas mejoras continuas en el campo tecnológico han provocado que el uso de la tecnología se vaya extendiendo y estandarizando para el tratamiento de personas con discapacidad, ya que su uso permite que estas personas puedan realizar tareas de manera mucho más sencilla y con un mayor nivel de autonomía, entre las que destacan el uso de tablets, aplicaciones,

o el uso de sistemas alternativos y aumentativos de comunicación (SAAC) [7].

Estos SAAC [8] están orientados a facilitar y complementar la comprensión del lenguaje hablado a personas con problemas en la comunicación. Para su funcionamiento suelen hacer uso de pictogramas, una de las técnicas de comunicación que han demostrado funcionar bien, ya que son una forma de representar tareas, objetos o conceptos, que les facilitan la forma de comunicarse con otras personas sin necesidad de hacer uso de palabras. Estos pictogramas deben de ser imágenes fácilmente comprensibles con tan solo una mirada, eliminando detalles que no favorecen al entendimiento. El uso de pictogramas permite a estas personas realizar tareas de manera mucho más sencilla que antes, ya sea para intentar comunicarse con otras personas, o para facilitar el seguimiento de sus rutinas, ya que de un vistazo son capaces de saber que actividad deben de realizar en ese momento. El uso de todas estas tecnologías tiene como objetivo principal ayudar a estas personas a ir desarrollando el habla y el entendimiento del lenguaje natural [9] [10].

Todos estos avances también deben ver reflejado su uso en el aula, parte fundamental en la enseñanza de estas personas, ya que es uno de los sitios donde más tiempo están. Es muy importante para estas personas estar en entornos tranquilos, adaptados para sus necesidades, y con ayudas para facilitar su enseñanza, ya que aumenta en gran medida su rendimiento académico [11]. Por lo que es muy importante hacer ver la importancia de la adaptación de los medios que utilizan, y promover la implementación de tecnología asistiva en la vida de estas personas.

3.2. Tecnologías relacionadas

Antes de comenzar con la fase de análisis del proyecto se va a realizar un pequeño estudio sobre los sistemas ya existentes orientados a cubrir necesidades y objetivos similares a este proyecto, o que implementen soluciones que puedan ser de utilidad o estar relacionadas con este proyecto.

Para comenzar con el estudio, se van a buscar aplicaciones y software que implementen soluciones similares, que puedan aportar conocimientos, y lleguen a ser de utilidad para el problema a resolver.

Existen multitud de aplicaciones móviles que dan soluciones a distintos problemas que están relacionados con el TEA. Una de estas aplicaciones es PictoTEA [12], la cual es una aplicación exclusiva del sistema operativo Android y se puede encontrar en la Google Play Store. Esta aplicación está diseñada para facilitar la comunicación a personas con TEA mediante el uso de pictogramas, ya que permite añadir pictogramas en diferentes etapas con distintos niveles de dificultad, lo que permite que la persona vaya avanzando en el aprendizaje y pueda ir utilizando nuevos pictogramas y categorías. La parte más interesante de esta aplicación es que recoge los pictogramas de una API externa y los muestra al usuario, facilitando en gran medida a los usuarios la búsqueda de nuevos pictogramas. Otra aplicación orientada a intentar aportar soluciones y proporcionar ayuda para personas con TEA es Dictapicto [13], una aplicación diseñada para dispositivos móviles, aunque en este caso está dispo-

nible tanto para Android como para iOS. Esta aplicación permite pasar un mensaje de voz o texto, a imágenes, por lo que también está orientada a la utilización de pictogramas para ayudar a las personas a mejorar la comunicación. En este caso puede llegar a ser interesante el concepto de búsqueda de pictogramas mediante voz, aunque ya existen soluciones integradas en los propios teclados de los smartphones, o en asistentes de sistemas operativos, y que ya pueden realizar esta funcionalidad sobre un buscador de pictogramas por texto existente.

En un concepto de aplicación un tanto distinto de las anteriores está la aplicación Agenda de Pictogramas [14]. Es una aplicación móvil disponible tanto para dispositivos con sistema operativo iOS como dispositivos Android, que permite establecer grupos de actividades, organizados por días, haciendo uso de pictogramas, es decir, asignar para cada día de la semana una serie de pictogramas ordenados. Esta aplicación tiene como objetivo el facilitar el seguimiento de una rutina de manera sencilla para el usuario. De cierta manera, esta aplicación permite crear listas de pictogramas dándole al usuario la posibilidad de acceder a un determinado día en el calendario y que se listen, por orden temporal, cada una de las actividades que realizará durante el día. Esto podría tener similitudes con el concepto de lista de pictogramas reproducible que se pretende implementar en el sistema, aunque en este caso no se tiene en cuenta el tiempo de realización de cada actividad y al usuario se le muestran todos los pictogramas juntos. Otra aplicación bastante similar a la anterior es PictogramAgenda [15], la cual es una aplicación móvil solo disponible para dispositivos con sistema operativo Android. Los objetivos de la aplicación son parecidos a los de Agenda de Pictogramas, pero existen ciertas diferencias que hacen que también se deba de tener en cuenta, ya que permite al usuario subir sus propios pictogramas y crear grupos que representen a las tareas que se quieren realizar. Una vez estén los pictogramas en la aplicación, el usuario puede decidir cual de ellos se quiere marcar como tarea principal mientras las otras tareas continúan visibles en la parte superior como pendientes. Una vez se da por finalizada la tarea, es el propio usuario el que debe manualmente seleccionar otro de los pictogramas para que se vuelva el activo.

Tras realizar un análisis de las aplicaciones y sistemas existentes, se puede observar como no hay ninguna que implemente todos los objetivos que se pretenden alcanzar con este sistema. La gran mayoría de aplicaciones se limitan a la búsqueda y visualización de pictogramas, los cuales solo pueden ser gestionados por el mismo usuario que está utilizando la aplicación y además desde el mismo dispositivo. Ninguna de ellas se centra en la gestión del tiempo, permitiendo asignarles duración e implementando el cambio automático entre pictogramas, o posibilitar la gestión por parte de otro usuario que no esté utilizando el sistema, objetivos fundamentales que debe alcanzar este proyecto.

En cuanto a estudios y publicaciones que puedan ayudar y apoyar la realización del proyecto, hay que destacar la publicación **Classroom design for living and learning with autism** [16] realizada por Clare L. Vogel, donde realiza un estudio en el que aporta estándares de diseño y soluciones pensados para personas con trastorno del espectro autista. Es una publicación de gran utilidad, ya que proporciona indicaciones para realizar diseños contrastados y funcionales, lo que facilita en gran

medida las guías de diseño iniciales del sistema como puede ser la necesidad de predictibilidad o la minimización de distracciones.

Otra de las publicaciones que trata temas relacionados es el artículo **Classroom-Based Assistive Technology: Collective Use of Interactive Visual Schedules by Students with Autism** [17], en donde se realiza un resumen de tecnologías asistivas que son utilizadas para ayudar a las personas con TEA, pero en este caso centradas en establecer horarios. En este artículo se habla de la importancia de los apoyos visuales como los pictogramas, y como estos pueden ayudar a estas personas a organizar sus propias actividades y horarios. En el artículo se destaca el uso del sistema **vSKED**, una tecnología asistiva interactiva y colaborativa, que presenta en un dispositivo las tareas asignadas a cada uno de los alumnos, permite a los alumnos interactuar con el sistema, y está completamente sincronizado con la parte de la profesora. Además, para presentar la información a los alumnos, hace uso de una pantalla principal de 42 pulgadas, en donde todos los alumnos pueden ver las tareas asignadas, y además, cada alumno tendrá un dispositivo propio de 7 pulgadas en donde también podrá interactuar con el sistema.

Es un dispositivo algo antiguo pero tiene el objetivo claro de pasar las herramientas que se utilizaban en papel a un formato electrónico. Es un sistema parecido al que se quiere implementar, ya que utiliza una arquitectura cliente-servidor, utiliza un dispositivo principal en donde se muestran las actividades, y la profesora puede controlarlo externamente desde un ordenador en tiempo real. Sin embargo, este sistema no tiene en cuenta la gestión del tiempo, careciendo de indicativos y delegando en la profesora o los alumnos marcar la finalización de las tareas.

Por otra parte, otro tipo de dispositivo cuyo objetivo está más orientado a ayudar a personas con dificultad para percibir el tiempo es **Time Timer** [18], un dispositivo pensado para representar el transcurso del tiempo de forma mucho más visual. Este tipo de dispositivo tiene la intención de mejorar las capacidades de percepción del tiempo en personas con dificultad para hacerlo. El uso de este dispositivo para contabilizar el tiempo de realización de tareas, puede hacer mejorar significativamente la capacidad de estas personas de predecir e interpretar el tiempo que le pueden dedicar a cada una de ellas [19]. De este dispositivo cabe destacar la estrategia que utiliza para hacer la representación del tiempo restante, utilizando el color rojo para representar el tiempo que falta para su finalización.

4

Especificación y Análisis de Requisitos

4.1. Introducción

Este capítulo se estructurará en base al estándar IEEE 830 [20] para definir un documento de Especificación de Requisitos Software (ERS) para el proyecto.

El propósito principal de este documento de análisis es definir y explicar los requisitos generales y específicos que debe cumplir el software del proyecto, para asegurar que su funcionamiento y calidad final sean los esperados.

4.1.1. Alcance

Tecnologías IoT aplicadas a la gestión del tiempo para personas con TEA está orientado a personas que sufren del Trastorno del Espectro Autista (TEA), ayudándoles a entender y organizar las actividades que realizan durante el día.

El sistema hará uso de un dispositivo capaz de mostrar pictogramas con las actividades que llevarán a cabo, así como un indicador lumínico que muestre el tiempo restante de la actividad.

Además, para poder gestionar el funcionamiento del sistema se hará uso de una aplicación que permita crear y controlar la reproducción de las distintas actividades.

4.1.2. Definición, siglas y abreviaciones

En la Tabla 4.1 se presentan las definiciones, siglas y abreviaciones que se utilizarán a lo largo del capítulo.

Término	Definición
Raspberry Pi	Ordenador de tamaño y coste reducido que permite realizar proyectos con espacios reducidos
Servidor	Parte de la arquitectura del sistema encargada de responder a las peticiones
Cliente	Parte del sistema que se encarga de mostrar la parte visual y realizar las peticiones de datos al servidor
ARASAAC [21]	Siglas del Centro Aragonés para la Comunicación Aumentativa y Alternativa, el cual proporciona una API que da acceso a los pictogramas
Pictograma	Signo de la escritura de figuras o símbolos que sirven para representar objetos reales.
TEA	Son las siglas de Trastorno del Espectro Autista.

Tabla 4.1: Esta es una tabla con las **definiciones** de los conceptos más importantes que aparecen en este documento.

4.1.3. Visión General

El presente capítulo de especificación de requisitos consta de tres secciones diferenciadas y se estructura de la siguiente manera.

En primer lugar se realiza una pequeña introducción del documento y del sistema. A continuación, en la segunda sección se realiza una descripción general del funcionamiento del sistema, donde se darán a conocer sus principales funciones, sus restricciones y sus dependencias, dando una descripción global del mismo sin entrar en detalles específicos.

Por último, en la última sección del documento se definen los requisitos funcionales y no funcionales del sistema en detalle, especificando lo que realizará el sistema.

4.2. Descripción General

Este apartado servirá para describir el contexto que afecta al sistema y a sus requisitos. Esto ayudará a entender los requisitos específicos del sistema que se describirán de forma detallada en el siguiente apartado.

4.2.1. Perspectiva del producto

El sistema parte desde cero, es decir, no tiene ninguna relación con otros productos o sistemas. Está formado por dos partes principales, la primera encargada de la gestión y control del sistema, y la segunda encargada de mostrar los pictogramas.

4.2.2. Funciones del producto

En la Figura 4.1 se presenta el diagrama de casos de uso del sistema. Como se puede observar existen dos tipos de usuarios que interactúan directamente con el sistema.

En primer lugar está el gestor, el cual se encarga de la gestión de listas, control de la reproducción, de mostrar los pictogramas, y de configurar las luces, mientras por el otro lado está ARASAAC, el cual interactúa con el sistema en la búsqueda de pictogramas y en la funcionalidad de añadir pictogramas a listas.

Todos los casos de uso del sistema se pueden ver a continuación en la lista, aunque para hacer el documento más legible, la descripción en detalle de todos estos casos de uso puede ser encontrada en el Anexo C.

- CU-01 Controlar Reproductor de Pictogramas
- CU-02 Reproducir Pictograma
- CU-03 Pausar Pictograma
- CU-04 Avanzar Pictograma
- CU-05 Anterior Pictograma
- CU-06 Modificar Tiempo
- CU-07 Gestión de Pictolistas
- CU-08 Crear Pictolista
- CU-09 Editar Pictolista
- CU-10 Eliminar Pictolista
- CU-11 Reproducir Pictolista
- CU-12 Detener Pictolista
- CU-13 Obtener Pictolistas
- CU-14 Buscar Pictograma
- CU-15 Añadir Pictograma
- CU-16 Mostrar Pictograma
- CU-17 Configurar Luces LEDs

4.2.3. Características de los usuarios

Existen dos perfiles de usuarios que utilizarán la aplicación, cada uno desde un apartado del sistema diferente.

En primer lugar está el usuario encargado de gestionar y controlar el sistema y la persona con TEA que aunque no interactúa directamente, visualizará las salidas del sistema.

En segundo lugar está ARASAAC, una API del Gobierno de Aragón en la que se pueden consultar y obtener los pictogramas más utilizados actualmente.

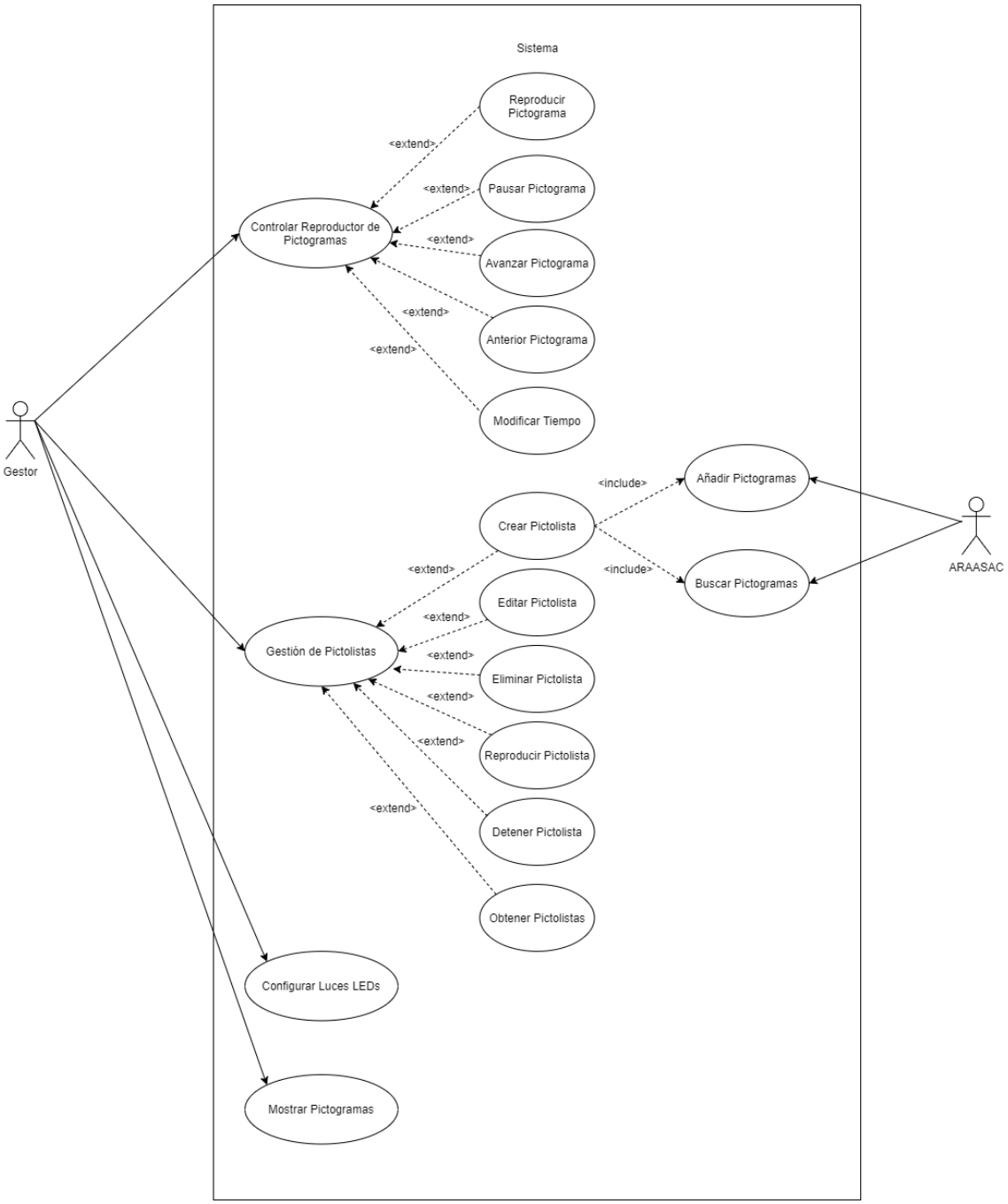

Figura 4.1: Diagrama de casos de uso del sistema

4.2.4. Restricciones

Para el desarrollo del sistema se han establecido ciertas limitaciones que se deben cumplir para poder obtener la calidad final deseada.

- Los pictogramas deben de ser fáciles de entender, para ello se usarán los pictogramas estándar más utilizados.
- El tiempo restante que le queda a la actividad actual se debe de indicar de la manera más visual y clara posible.
- La aplicación de control debe ser fácilmente accesible y funcional desde dispositivos móviles y ordenadores.
- El dispositivo debe ser pequeño y resistente a caídas.
- El sistema debe tener una interfaz sencilla de entender y manejar para usuarios poco experimentados.
- Se deberá permitir el control concurrente de la aplicación desde varios dispositivos diferentes.

4.2.5. Atenciones y dependencias

El sistema está diseñado para no tener que depender de un hardware específico, aunque para el funcionamiento de la tira de luces LED si que sería necesario contar con una Raspberry Pi o similares.

Para que se puedan seguir viendo los pictogramas, será necesario que la API de ARASAAC [22] continúe funcionando correctamente y sirviendo los datos de la misma manera que cuando se ha diseñado el sistema.

4.2.6. Requisitos Futuros

Se podría dar el caso de que la API de ARASAAC [22] sufriera cambios en su implementación y fuera necesario crear un nuevo diseño para su interacción e integración.

Se entiende que el resto de requisitos que aparecen en este documento son estables y no se producirán cambios en el futuro.

4.3. Requisitos Específicos

4.3.1. Interfaces Externas

- La **Interfaz de usuario** de la aplicación debe ser fácil de entender y de utilizar. Su uso principal se hará desde dispositivos móviles, pero al ser multiplataforma también puede ser utilizada desde un ordenador o tableta. Esto por tanto condiciona también la interfaz, ya que debe estar principalmente orientada al uso táctil móvil pero manteniendo su estructura y diseño de igual forma en otros dispositivo con pantallas más grandes.
- **Interfaz hardware**: Para la utilización de las luces LEDs es necesaria la utilización de una de un dispositivo hardware que permita conectar y gestionar su funcionamiento. Si no se necesita utilizar las luces LED, también es posible hacer uso de dos dispositivos, uno que funcione como controlador y otro encargado de mostrar los pictogramas.

- Existen tres **interfaces de comunicación** en el sistema:
  - Para la comunicación del sistema entre la aplicación cliente y el servidor con respecto al envío y peticiones de datos al sistema de almacenamiento, se utilizan protocolos web HTTP con cargas útiles JSON.
  - Para la comunicación entre los distintos módulos de la aplicación se utilizan WebSockets, ya que permiten establecer comunicaciones en tiempo real y manteniendo una comunicación constante.
  - Por último para realizar la búsqueda de pictogramas y obtener las imágenes el servidor realiza peticiones HTTP a la API de ARASAAC [22].

4.3.2. Requisitos Funcionales

En esta sección se definirán los requisitos funcionales del sistema. Estos requisitos funcionales son aquellos que hacen referencia al comportamiento de las funciones y servicios del sistema. Son necesarios para comenzar con la fase de diseño, ya que con ellos se define la estructura y el alcance del proyecto que se va a desarrollar.

Estos requisitos se han obtenido mediante la realización de estudios de trabajos relacionados, tras haber mantenido conversaciones informales con el tutor, y tras haber realizado entrevistas informales con expertos en la cuestión. Estas entrevistas informales tuvieron lugar en **Aleta** [23], un centro especialista en el trabajo con personas con TEA y que hacen un uso masivo de la tecnología en el aula. Allí se discutió la idea y se presentaron algunas propuestas a la directora del centro, muchas de las cuales se han visto reflejadas en este análisis y requisitos obtenidos.

Los requisitos funcionales que debe de tener el sistema son los que se listan a continuación, aunque su descripción detallada puede ser encontrada en el Anexo D del presente documento.

- RF01 Estado del sistema
- RF02 Sincronización entre dispositivos
- RF03 Reproducir lista
- RF04 Pausar lista
- RF05 Avanzar pictograma
- RF06 Retroceder pictograma
- RF07 Modificar tiempo de reproducción
- RF08 Consultar pictolistas
- RF09 Crear pictolistas
- RF10 Eliminar pictolistas
- RF11 Modificar pictolistas
- RF12 Buscar pictogramas
- RF13 Conexión a ARASAAC
- RF14 Añadir pictogramas
- RF15 Borrar pictogramas

- RF16 Modificar pictogramas
- RF17 Asignar tiempo a un pictograma
- RF18 Seleccionar pictolista
- RF19 Parar pictolista
- RF20 Personalizar LEDs
- RF21 Representación LEDs

4.3.3. Requisitos No Funcionales

Los requisitos no funcionales son aquellos requisitos que imponen restricciones en el diseño o la implementación del sistema. Estas pueden ser restricciones en el tiempo, sobre estándares en el desarrollo o implementación, rendimiento del sistema o seguridad entre otras [24].

De la misma forma que los requisitos funcionales, estos han sido obtenidos mediante conversaciones informales con el tutor, expertos en el tema y el estudio del estado del arte.

Los requisitos no funcionales que debe cumplir el sistema son los que se listan a continuación, aunque su descripción detallada puede ser encontrada en el Anexo E del presente documento.

- RNF01 Diseño IU
- RNF02 Soporte multiplataforma
- RNF03 Velocidad de respuesta
- RNF04 Soporte multidioma
- RNF05 Persistencia de la información
- RNF06 Recursos hardware
- RNF07 Navegación por la interfaz
- RNF08 Obtención de pictogramas
- RNF09 Sistema modular
- RNF10 Personalización
- RNF11 Seguridad
- RNF12 Mostrar errores
- RNF13 Uso de colores

4.3.4. Requisitos del Diseño

El diseño del sistema deberá estar preparado para ser ejecutado en hardware de bajo coste como la Raspberry Pi [25], además, para la correcta comunicación entre la aplicación y el hardware será necesario hacer uso de la arquitectura cliente-servidor.

4.3.5. Atributos del Sistema

Existen cuatro atributos de calidad que se pueden detallar del sistema. Estos son la seguridad, el mantenimiento, la fiabilidad y la portabilidad del mismo.

- La **seguridad** de la aplicación no debe permitir la conexión de dispositivos externos a la red WiFi a la que está conectado. Como no existen usuarios con distintos niveles de permisos no sería necesario una pantalla de acceso, la cual solo perjudicaría la usabilidad y accesibilidad. Por ello, la seguridad de la propia red WiFi es la que va a asegurar que el sistema no se vea comprometido.
- El **mantenimiento** del sistema es muy sencillo, ya que su diseño modular permite añadir nuevas funcionalidades o mejoras de las ya existentes de manera fácil.
- Con respecto a la **fiabilidad** del sistema depende en gran medida de disponer de una red local con una conexión estable, ya que la comunicación entre la aplicación y el hardware se realiza de esta forma. Además el hardware, a pesar de ser resistente se puede ver comprometido en caso de ser sometido a grandes cargas físicas.
- La **portabilidad** del sistema es muy alta, ya que el diseño y las tecnologías que se utilizan le permiten ser ejecutado en hardware de bajo coste y en múltiples plataformas, incluyendo smartphones, ordenadores y tabletas, tanto en formato web como de aplicación.

4.3.6. Matriz de Trazabilidad

La matriz de trazabilidad sirve para poder relacionar los requisitos funcionales con los casos de uso. La matriz de trazabilidad resultante es la que se puede ver en la Figura 4.2 en la cual se puede observar como todos los requisitos funcionales están cubiertos por al menos un caso de uso.

RF/CU	RF01	RF02	RF03	RF04	RF05	RF06	RF07	RF08	RF09	RF10	RF11	RF12	RF13	RF14	RF15	RF16	RF17	RF18	RF19	RF20	RF21
CU-01	X	X	X	X	X	X	X						X								X
CU-02			X																		
CU-03				X																	
CU-04					X																
CU-05						X															
CU-06							X														
CU-07	X	X						X	X	X	X	X	X	X	X	X	X	X	X		
CU-08									X												
CU-09											X				X	X	X				
CU-10										X											
CU-11																		X			
CU-12																			X		
CU-13								X													
CU-14												X	X								
CU-15														X	X	X	X				
CU-16													X								X
CU-17	X	X																		X	

Figura 4.2: Matriz de trazabilidad

5

Diseño

Tras la fase de Especificación y Análisis de Requisitos se continuará con la fase de diseño del sistema. Este capítulo se dividirá en las tres partes que componen al sistema, el diseño del software, el diseño del hardware, y por último la arquitectura del sistema que los contiene a los dos.

5.1. Diseño Hardware

Para cumplir con los requisitos del sistema definidos en las secciones anteriores de Requisitos Funcionales y Requisitos No Funcionales, es necesario un diseño hardware con las siguientes características:

- Es necesario que el hardware disponga de una **pantalla LCD** que permita mostrar los pictogramas seleccionados por el usuario desde la aplicación de control.
- Para cumplir con los requisitos funcionales RF20 y RF21, debe disponer de una tira de **luces LED** que permita representar el tiempo que le resta al pictograma que se está mostrando actualmente.

La primera decisión con respecto al hardware sería el tipo de dispositivo que se va a utilizar para la implementación del proyecto, para este caso, debido a las características anteriormente mencionadas, y teniendo en cuenta el cumplimiento del requisito no funcional RNF06, la mejor opción es claramente la utilización de una Raspberry Pi [25].

Otros dispositivos como Arduino [26] o placas de desarrollo del tipo Wemos D1 Mini [27] no podrían encargarse de la ejecución del servidor y de la utilización de pantallas LCD para mostrar los pictogramas, por lo que solo servirían para el control de las luces LED y obligarían a utilizar otros dispositivos que sumarían mas hardware y costes al proyecto. Sin embargo, la Raspberry Pi permite ejecutar sistemas operativos de escritorio con interfaces gráficas, tiene una mayor potencia, tiene salidas de video HDMI (lo que permite conectarle cualquier pantalla), da la posibilidad de instalar aplicaciones y servicios web, y además permite conectar componentes a sus pines GPIO, por lo que es una opción mucho más completa y que a pesar de ser un poco más cara, permite realizar todo con un único dispositivo.

Una vez definidas las características que debe cumplir el hardware, el prototipo en papel aproxima-

do del dispositivo que se quiere desarrollar puede ser parecido al que vemos en la Figura 5.1, en la cual podemos ver la pantalla, en donde se puede ver el pictograma, conectada a una tira de luces LED. Como se puede observar, el objetivo principal es el de conseguir un dispositivo que cuente con una pantalla y una tira de luces LED integradas, que pueda ser ensamblado de manera sencilla, y ocupe poco espacio.

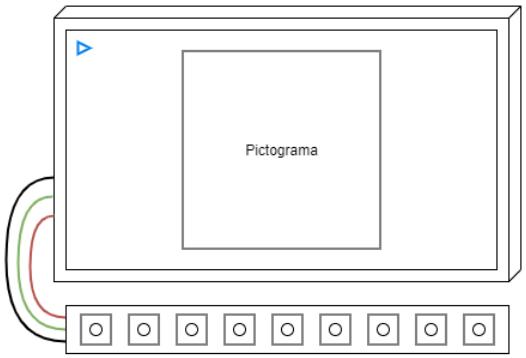

Figura 5.1: Prototipo en papel del dispositivo

Junto a la Raspberry Pi, existen ciertos componentes que son necesarios para la consecución de los objetivos del proyecto, como pueden ser las tiras de luces LED, o las pantallas donde se mostrarán los pictogramas. Para cada uno de los componentes necesarios existen múltiples opciones en el mercado, las cuales se analizarán a continuación y facilitará la tarea de escoger la mejor opción.

Pantallas

Existen varias opciones que pueden servir como pantalla para mostrar los pictogramas en la Raspberry Pi, aunque destacan por encima del resto dos variantes principales, las pantallas de tinta o ePaper, y las pantallas de tipo LCD. Las características principales de cada tipo de pantalla son las que se enumerarán a continuación.

Las **pantallas de tinta** muestran las imágenes en blanco y negro, tienen una tasa de refresco baja, y se parecen más a lo que puede ser un dibujo en papel. Una de sus grandes ventajas es su flexibilidad junto a su bajo consumo, ya que solo gastan energía en los cambios de imagen. Su tamaño suele ser pequeño, por lo que hay que estar más cerca para verla. Un ejemplo de pantalla de tinta compatible

con la Raspberry Pi es la que se puede observar en la Figura 5.2, en donde se puede apreciar como el tamaño de la pantalla es más pequeño pero mucho más flexible y fácil de colocar.

Figura 5.2: Ejemplo de una pantalla ePaper

Por otro lado están las **pantallas LCD**, las cuales se parecen más a las pantallas tradicionales de los ordenadores o móviles. Este tipo de pantallas si muestran colores y su tasa de refresco es muy rápida, a cambio su consumo es más elevado. Se puede ver un ejemplo de este tipo de pantallas para Raspberry Pi en la Figura 5.3, en donde es apreciable que su tamaño es más grande pero en cambio es más difícil de colocar debido a su rigidez.

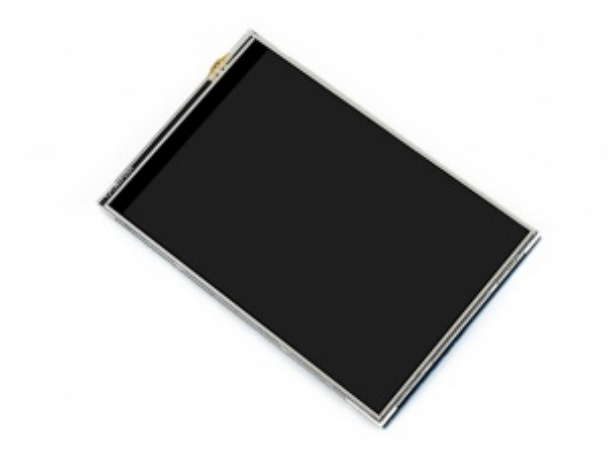

Figura 5.3: Ejemplo de una pantalla LCD

Entre las dos opciones, la pantalla de tinta tiene un tamaño demasiado pequeño, lo que obligaría a acercarse demasiado al dispositivo, además al no tener color puede ser más complicado entender los pictogramas. Por otra parte, las desventajas de la pantalla LCD como el mayor consumo, no supondrían un gran inconveniente, ya que el dispositivo va enchufado directamente a la corriente.

Luces LED

La elección de componentes en la parte de la tira de luces LED es más sencilla, ya que los requisitos que deben de cumplir son, que sea compatible con la Raspberry Pi, que sea una tira de luces RGB, y que cada uno de los LEDs pueda ser controlado de manera individual.

Para este caso existen dos opciones que pueden ser utilizadas con resultados parecidos. La primera opción consistiría en la utilización de luces LED RGB individuales, lo cual obligaría a realizar la conexión entre cada una de ellas y la Raspberry Pi, por lo que también se limitaría el número de luces que se pueden utilizar. Como segunda opción se podrían utilizar tiras de luces LED RGB ya ensambladas por un fabricante, lo que permite el uso de señales de reloj para controlarlas y por tanto menos conexiones con la Raspberry Pi.

Un ejemplo de este tipo de tiras LED puede verse en la Figura 5.4, las cuales están todas unidas entre sí y disponen de tres conectores del tipo JST PH, y que en este caso corresponde con un modelo de Adafruit [28].

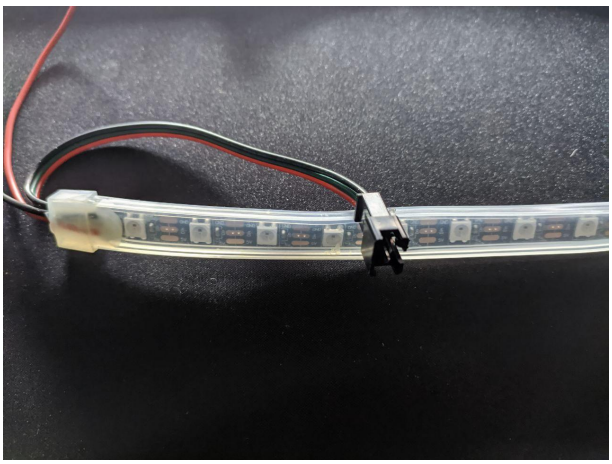

Figura 5.4: Ejemplo de tira de LEDs

Por tanto, una vez analizadas las dos opciones, se ha llegado a la conclusión de que la utilización de tiras LED es mejor para este proyecto, ya que simplifica mucho el cableado, el control de las luces se puede hacer desde una librería de terceros del fabricante, y la colocación en el dispositivo final es más sencilla.

5.2. Diseño Software

Para el correcto diseño del software hay que tener en cuenta varios factores que van a condicionar su implementación, y que tendrán como finalidad cumplir los requisitos anteriormente descritos en la fase de análisis.

El primero de estos factores es el tipo de aplicación que se va a realizar, ya que existen múltiples sistemas operativos que podrían utilizarla. Uno de los requisitos anteriormente mencionados, en concreto el requisito no funcional RNF02, comentaba que la aplicación debía de ser multiplataforma, por lo que para evitar el desarrollo específico para cada sistema operativo la mejor decisión pasa por implementar una aplicación web, ya que esto permite que pueda ser accedida desde cualquier dispositivo con un navegador actualizado.

Para que la aplicación se vea bien en cualquier dispositivo, debe implementar una interfaz RESPONSIVE que se adapte a cualquier tamaño de pantalla y resolución, ya sea la pantalla de un dispositivo móvil, de una tableta, o de un ordenador de escritorio. Esta implementación debe respetar el requisito RNF01, por lo que es importante saber que aunque el acceso más común se realizará mediante dispositivos táctiles, no se debe descuidar el diseño para otros tamaños de pantalla.

Otro de los puntos claves es la compatibilidad con los distintos navegadores y sus versiones. Esta decisión no vendrá condicionada por la versión de los navegadores de escritorio que se utilizan en los ordenadores, ya que estos son fácilmente actualizables o reemplazables por otros. El problema principal de las versiones de navegador viene por la versión de los sistemas operativos móviles, puesto que estos últimos son los que implementan los navegadores nativos que ejecutarán las aplicaciones, y a pesar de que si podrían acceder como si fuera una página web, no la podrían utilizar como una aplicación nativa implementada mediante Webviews.

Para solucionar estos problemas, la mejor solución pasaría por la utilización de entornos de desarrollo como Apache Cordova [29], los cuales permiten el desarrollo de aplicaciones multiplataforma (móviles, escritorio, navegador, etc) mediante código HTML, CSS y Javascript. Este tipo de marcos de desarrollo permite unificar esfuerzos, permitiendo con un código único llegar a una gran cantidad de dispositivos, plataformas y sistemas operativos, facilitando también la compatibilidad con múltiples versiones de sistemas operativos o navegadores.

Esta configuración permite que una vez esté generado el código, este pueda ser construido para múltiples plataformas, en el caso de esta aplicación, se generará un código principal que será accedido desde un navegador web y que se ejecutará desde un servidor web en la Raspbnerry Pi. Alternativamente se podrá generar también aplicaciones para su uso en Android, iOS o incluso como aplicación de escritorio.

5.2.1. Prototipos Aplicación

En esta subsección se mostrarán los prototipos en papel de las vistas de las que dispondrá la aplicación, junto a una pequeña descripción de cada una de ellas. Estos prototipos de las vistas de la aplicación han sido diseñados para el cumplimiento de todos los requisitos funcionales y no funcionales recogidos en el apartado anterior.

En la Figura 5.5, se puede ver la pantalla principal de la aplicación. Es la primera que aparece al abrir la aplicación y sirve para controlar la reproducción de los pictogramas.

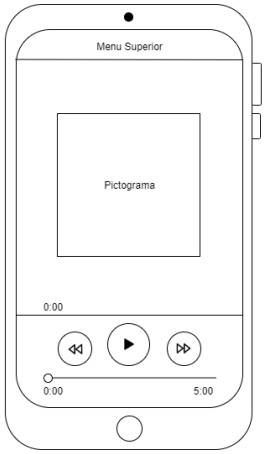

Figura 5.5: Imagen prototipo del reproductor

En la Figura 5.6, se puede ver el menú en donde se listarán todas las pictolistas creadas. Desde esta vista se permitirá acceder a ellas para modificarlas, eliminarlas, o crear nuevas listas.

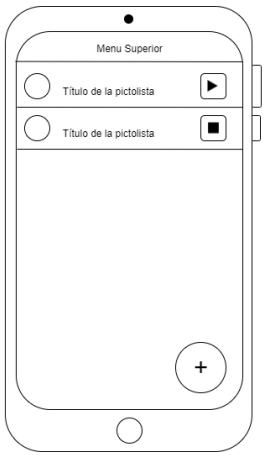

Figura 5.6: Imagen prototipo del listado de pictolistas

En la Figura 5.7 se puede ver el menú desde donde se puede realizar la búsquedas de pictogramas mediante texto y seleccionarlos para añadirlos a las listas.

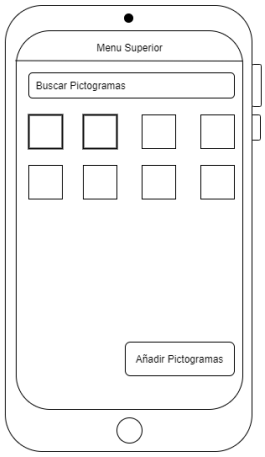

Figura 5.7: Imagen prototipo de la búsqueda de pictogramas

En la Figura 5.8 se puede ver la pantalla desde donde se introducen los datos para la creación o edición de pictolistas.

Figura 5.8: Imagen prototipo de la creación de pictolistas

En la Figura 5.9 se puede observar la vista desde donde se pueden modificar algunos ajustes de la aplicación y realizar la personalización de las luces LED.

En la Figura 5.10 se puede observar el prototipo de la pantalla correspondiente al visor de pictogramas, en donde se puede ver en la parte superior izquierda un indicativo del estado de la reproducción, y en el centro el pictograma que se está reproduciendo.

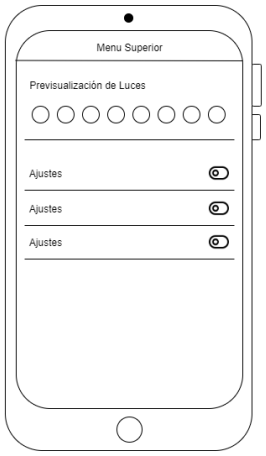

Figura 5.9: Imagen prototipo de la vista de ajustes

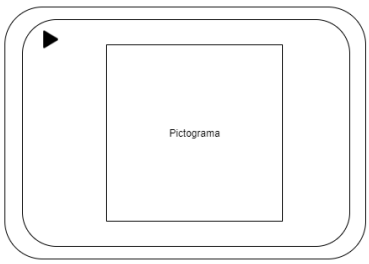

Figura 5.10: Imagen prototipo del visor de pictogramas

5.3. Arquitectura del Sistema

Tras realizar el análisis y diseño de sistema, es necesario definir el tipo de arquitectura que se adapta mejor a las necesidades del mismo y cumpla con los requisitos y objetivos marcados.

Para este caso lo más adecuado sería la utilización de una arquitectura cliente-servidor, tal y como se muestra en la Figura 5.11, donde existen dos partes claramente diferenciadas. En la parte de cliente, la cual es la que se encarga de realizar las peticiones, tenemos a la aplicación de gestión, mientras que en la parte del servidor será la API NodeJS [30] la que reciba esas peticiones.

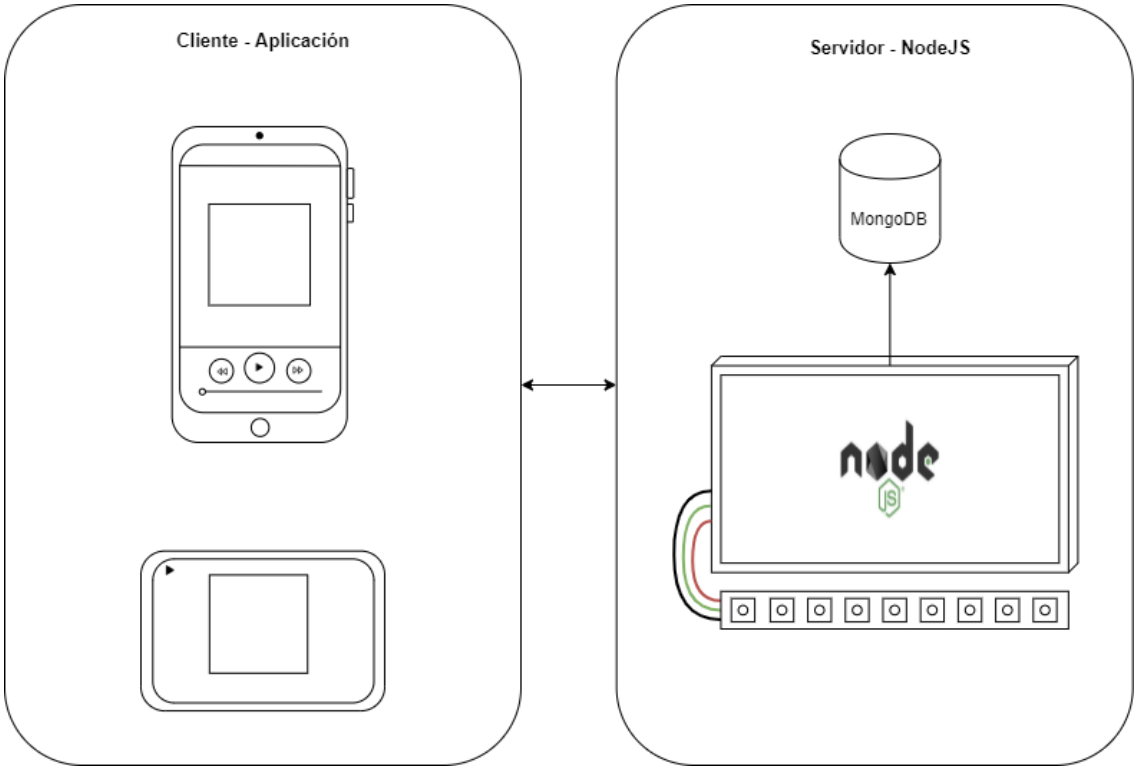

Figura 5.11: Arquitectura del Sistema

A continuación se detallará cada una de estas partes junto a cada una de sus funciones:

Cliente: Es la parte de la arquitectura que se muestra al usuario, lo que se suele conocer como el FRONTEND. Su función principal es la de permitir al usuario interactuar con el sistema y realizar las peticiones necesarias para interactuar con los datos, ya sea recogiéndolos, guardándolos, eliminándolos o modificándolos.

Servidor: La parte servidor es aquella encargada de recibir todas las peticiones que realizan los clientes para llevar a cabo las acciones y devolverles una respuesta. En el caso de este sistema se ejecuta directamente sobre el dispositivo y es el encargado de interactuar con la base de datos. Como es la parte que siempre se encuentra en el dispositivo, también

es el encargado de interactuar con las luces LED, controlándolas dependiendo de la configuración que el usuario le envíe desde la aplicación. Por último, al ser donde todos los dispositivos y usuarios se conectan, es el encargado de difundir a todos ellos el estado del sistema, asegurándose que entre ellos la información que se muestra es la misma y no existen problemas de sincronización entre dispositivos.

Dentro de la aplicación de gestión también existe una arquitectura software que tendrá dos partes claramente diferenciadas, tal y como se muestra en la Figura 5.12, en donde se pueden distinguir la parte del controlador y la parte del visor contenidas en la misma aplicación. Estas dos partes comparten datos en tiempo real entre si aunque sean ejecutadas desde distintos dispositivos, tal y como se pide en el requisito funcional RF02 recogido en la fase de análisis.

La primera parte consiste en la funcionalidad de gestión y control, donde el usuario puede crear listas, controlar la reproducción de pictogramas y personalizar las luces, mientras que la segunda parte es la encargada de mostrar los pictogramas y de reproducir las listas. Esta última parte es la que normalmente será ejecutada en la pantalla del dispositivo, mientras que la parte de control y gestión será a la que se acceda normalmente desde el dispositivo del usuario.

Como se puede ver en la Figura 5.12 la aplicación dispone de dos partes distintas, tal y como se explicó anteriormente. A continuación se detallará la función de cada una de ellas, como interactúan los dispositivos con ellas, y finalmente como se comunican entre ellas:

Controlador: Es la parte de la aplicación a la que accede por defecto el usuario desde sus dispositivos. Desde aquí el usuario puede personalizar las luces, crear nuevas listas, reproducirlas y finalmente controlar la reproducción. Desde el controlador se realizan peticiones al servidor NodeJS para que interactúe con la base de datos, almacenando, creando, modificando o eliminando nuevas listas.

Visor Pictogramas: Es la otra parte de la aplicación, es la encargada de mostrar los pictogramas que se están reproduciendo y que está diseñada para lanzarse automáticamente en la pantalla del dispositivo.

Para que estas dos partes compartan información en tiempo real, como por ejemplo, que el visor de pictogramas muestre el siguiente pictograma de la lista cuando el usuario avance la reproducción, será necesario que se comuniquen y reciban información del servidor lo más velozmente posible. Esta arquitectura permite que aunque sean dos partes distintas, donde a una accede el dispositivo del usuario y a la otra una Raspberry Pi, compartan tanto la estructura de la aplicación como los servicios utilizados para comunicarse y recibir información del servidor.

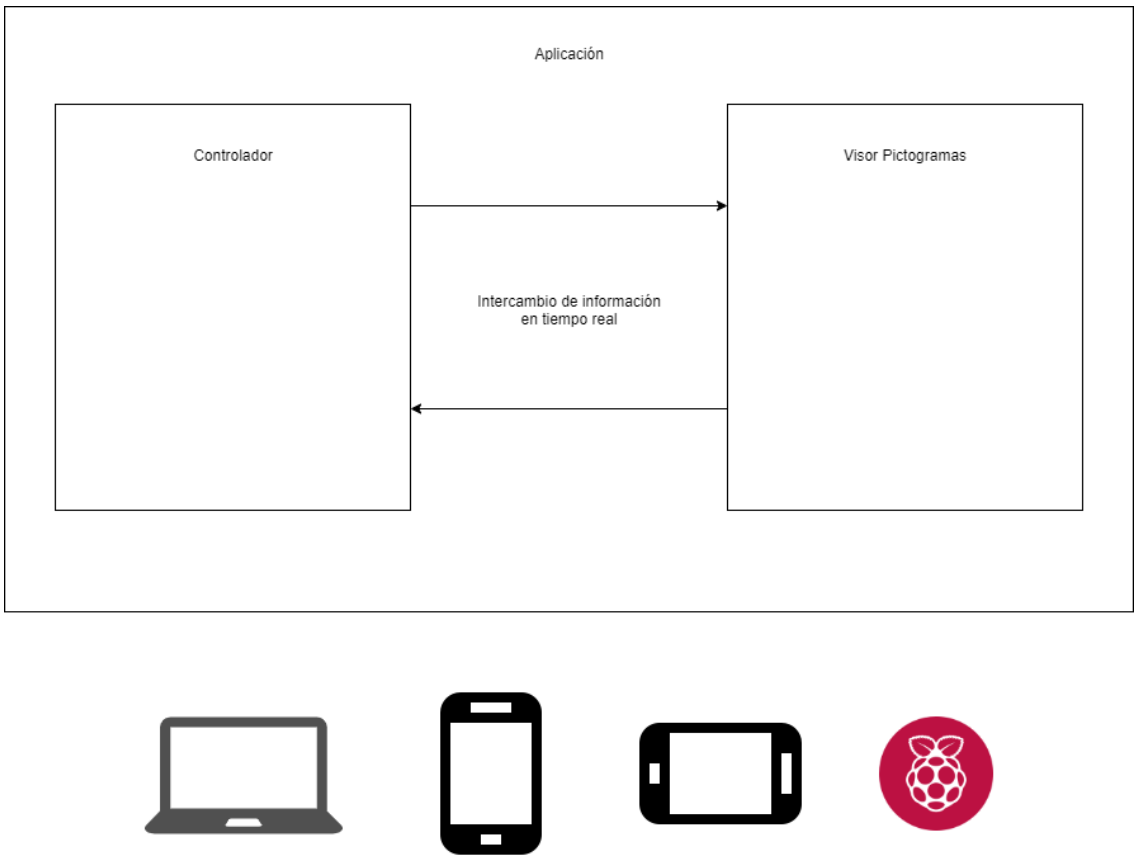

Figura 5.12: Arquitectura de la Aplicación

5.3.1. Diagramas de Secuencia

A continuación se mostrarán los diagramas de secuencia del sistema más importantes. El primero de ellos será el diagrama de secuencia general, el cual mostrará la interacción del sistema completo, mientras que los siguientes se centrarán más en partes concretas del mismo.

En la Figura 5.13 se puede ver el diagrama de secuencia general del sistema, en el cual se puede apreciar la interacción general de un usuario con el sistema en varias situaciones. En primer lugar el usuario realiza una acción sobre la aplicación de gestión, lo que genera una petición al servidor y provoca que este interactúe con las luces o la base de datos. Por último, el servidor devuelve una respuesta a la aplicación de gestión, en donde se interpretará y se mostrará un mensaje al usuario.

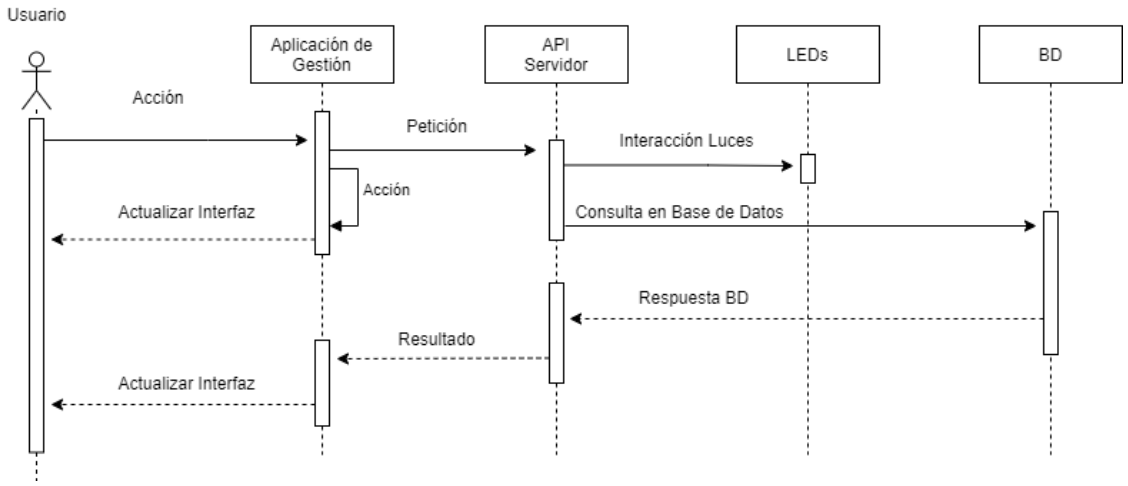

Figura 5.13: Diagrama de Secuencia General

En la Figura 5.14 se puede observar el diagrama de secuencia del control del reproductor, el cual representa la interacción del usuario con el sistema en las acciones relativas al control de la reproducción de pictogramas. En este caso el usuario realiza una acción sobre la aplicación, como puede ser pausar una lista, a continuación actualiza su propia interfaz, y envía esta misma acción al servidor. Esta acción tiene que ser enviada al servidor por dos motivos, para que el servidor envíe esta actualización al resto de dispositivos y para que modifique el comportamiento de las luces LED.

La Figura 5.15 representa el diagrama de secuencia de la gestión de pictolistas, en el cual se explica la interacción del usuario con el sistema en todas las acciones relacionadas con las listas. En primer lugar el usuario interactúa con la aplicación de gestión con acciones como crear o consultar listas, lo que provoca que se envíe esta petición al servidor. Una vez en el servidor, dependiendo de la acción realizada, este interactúa con la base de datos ya sea para insertar datos o para consultarlos. Una vez obtenidos, estos datos se envían de vuelta a la aplicación para que sean mostrados o confirmar la modificación de los datos al usuario.

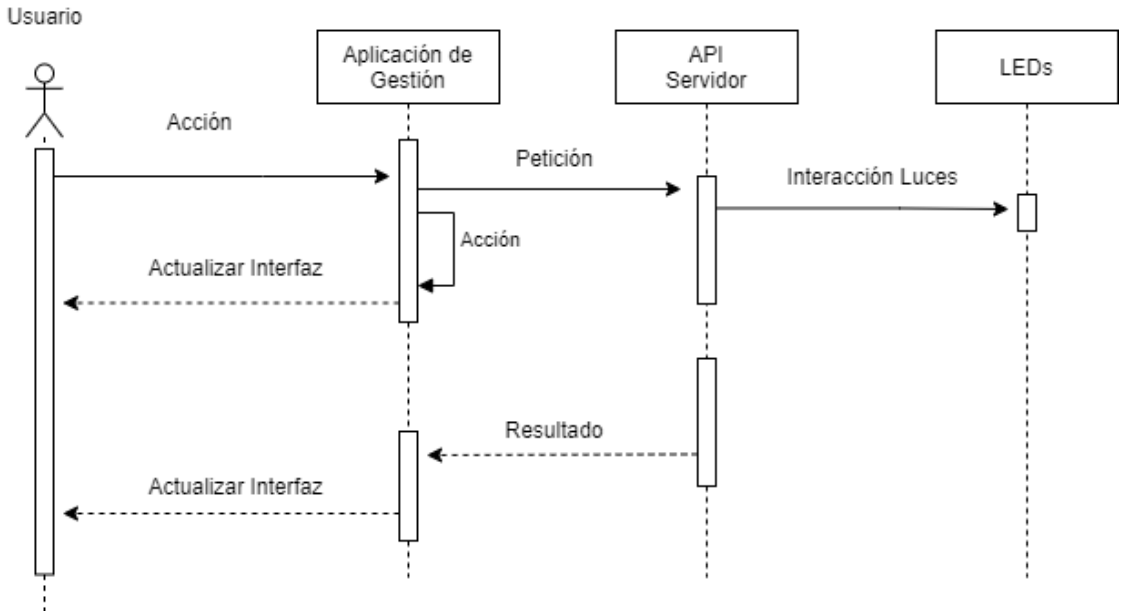

Figura 5.14: Diagramas de Secuencia Control del Reproductor

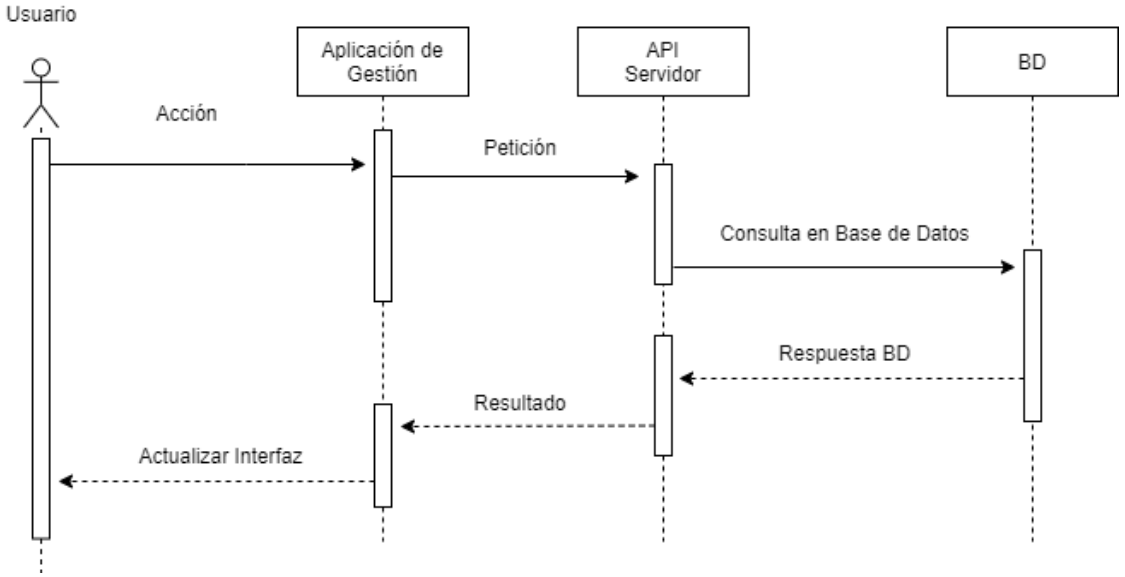

Figura 5.15: Diagramas de Secuencia Gestión de Pictolistas

En la Figura 5.16 está el diagrama de secuencia de la personalización de los LEDs, en donde se puede ver la interacción del usuario con el sistema cuando se quiere personalizar las luces LED. En este caso el usuario realiza un cambio en la configuración de las luces desde la aplicación de gestión, lo que provoca que esta nueva configuración se guarde en el LocalStorage del dispositivo y una vez confirmado se actualice la interfaz con la nueva configuración. Esto es debido a que este tipo de configuración permite que cada usuario guarde su configuración propia.

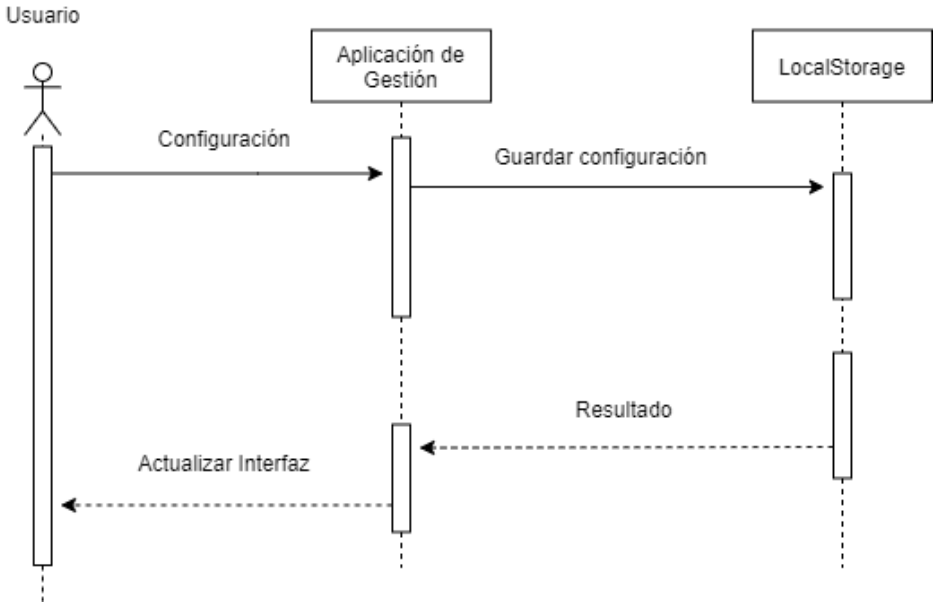

Figura 5.16: Diagramas de Secuencia Personalización LEDs

Por último, en la Figura 5.17 se puede apreciar el diagrama de secuencia de mostrar pictogramas, en donde se explica en detalle la interacción entre el visor de pictogramas y el usuario. En primer lugar el usuario comienza la reproducción de un pictograma desde la aplicación de gestión, lo cual provoca la petición al servidor y que la propagará a los distintos dispositivos conectados, entre los cuales también se encontrará el Visor de Pictogramas. Al Visor de Pictogramas le llegará la petición de comenzar la reproducción de un Pictograma, por lo que para obtenerlo, primero realizará una petición a la API de ARASAAC para que le devuelva el pictograma requerido y poder mostrarlo en pantalla al usuario.

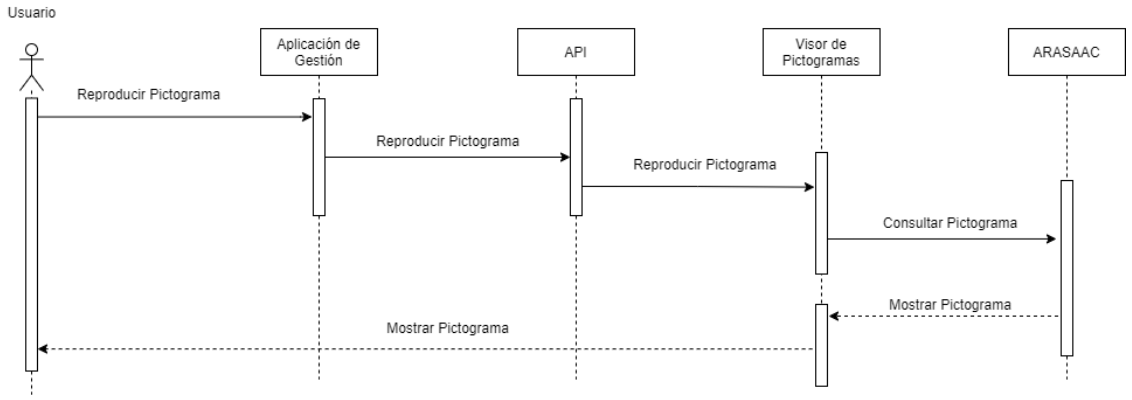

Figura 5.17: Diagramas de Secuencia Mostrar Pictograma

5.4. Modelado de Datos

Tras la fase de diseño del sistema realizado en los apartado anteriores es necesario diseñar también el modelo de datos, ya que este debe cumplir con los requisitos y objetivos descritos en la fase de Análisis.

Los requisitos que debe de poder cubrir el modelo de los datos para un correcto funcionamiento del sistema son los siguientes:

- La elección de los pictogramas debe ser sencilla, evitando la subida continua de imágenes y pictogramas por parte del usuario, tal y como se indican en los requisitos RF12 y RNF08.
- Se podrán crear varias listas distintas con distintos pictogramas en cada una y pudiendo cambiar rápido entre ellas, tal como se puede ver en los requisitos RNF05 y RF09.
- Cada pictograma debe tener asociado un tiempo de reproducción y que pueda ser diferente en cada lista, tal y como se indica en el requisito RF17.

Para la realización de este proyecto es necesario separar los datos en dos conceptos distintos, las pictolistas y los pictogramas, los cuales tendrán que ser almacenados por el sistema y se detallan a continuación.

Pictolistas

Una **Pictolista** es una agrupación de pictogramas que crea el usuario y que representa a la secuencia de actividades que realizarán las personas con TEA. Los atributos que hay que guardar sobre las pictolistas son los siguientes:

name: Cadena de caracteres que sirve para identificar una lista. El usuario podrá introducir el nombre que le permita diferenciarla de otras listas, este se mostrará en en listado.

description: Cadena de caracteres que representa una pequeña descripción de la pictolista. Esta descripción se mostrará junto al nombre en el listado.

Pictogramas

Los **Pictogramas** son las imágenes que el usuario quiere mostrar en el visor, las cuales representan conceptualmente a objetos o significados a los que se refieren. Los atributos que hay que almacenar con respecto a los pictogramas son los siguientes:

img: Cadena de caracteres que guarda el número de identificación de un pictograma. Este número de identificación es el utilizado por ARASAAC para diferenciar los pictogramas de los que disponen, por lo que puede utilizar para acceder a él utilizando la API de ARASAAC en cualquier momento.

time: Número entero que representa, en segundos, el tiempo que el pictograma se mostrará en el visor. Este es el tiempo base por defecto, dentro de la reproducción el usuario podrá pausarlo o modificarlo.

Configuración LEDs

Otro grupo de datos que tendrán que ser almacenados son la configuración de las luces LED que se realiza desde la pantalla de ajustes. Estos datos podrían ser almacenados de igual manera que el resto en la base de datos junto a las pictolistas y pictogramas, pero debido a que la configuración de las luces es única en el dispositivo, y puede ser un ajuste que se quiera modificar dependiendo del usuario que utilice el sistema, se pueden utilizar otras soluciones que no incluyan a la base de datos.

Para este caso, la mejor opción pasaría por utilizar el almacenamiento local de los dispositivos, ya que es una forma sencilla de almacenar las configuraciones de las luces cada vez que alguien las modifica. Esta forma de almacenar las configuraciones tiene sus ventajas con respecto al almacenamiento en la base de datos.

La mayor ventaja de este tipo de almacenamiento, es que al guardarse en el dispositivo que utiliza el usuario cada configuración de luces se queda almacenada para cada uno de ellos, por lo que al no compartir el mismo medio de almacenamiento entre todos, se permite guardar una configuración de personalización por dispositivo. Precisamente esta también es una de sus desventajas, ya que al realizar el almacenamiento por dispositivo, el usuario que accede desde distintos terminales tendría que configurar los ajustes de luces para cada uno de ellos. Sin embargo, para este caso las ventajas superan a las desventajas y es una mejor opción de implementación.

6 Implementación

Una vez realizadas las fases de análisis y diseño se puede comenzar con la fase de implementación y desarrollo del sistema. A lo largo de este capítulo se explicarán las elecciones de las distintas tecnologías empleadas y los puntos principales de implementación que se dan en cada uno de los componentes del sistema.

6.1. Base de Datos

Una vez definido y diseñado el modelo de datos necesario, se debe elegir el sistema de almacenamiento de datos que mejor se adapte al sistema definido.

La primera cuestión que se plantea es si se debe de utilizar una base de datos relacional SQL, o sin embargo, un sistema no relacional NoSQL se adapta mejor al diseño de este proyecto. Cada uno de los sistemas tiene sus ventajas y desventajas [31]:

Sistema Relacional SQL

- Ventajas
  - Mucho mas estable y maduro, ya que lleva mucho tiempo siendo el sistema más utilizado, y por tanto existe mucha información a disposición de los desarrolladores.
  - Estándares de diseño y creación muy claros y bien definidos, gracias a la utilización del lenguaje SQL.
  - Buenas herramientas e interfaces gráficas para el diseño y gestión de la base de datos.
- Desventajas
  - La administración y mantenimiento del sistema es un poco más costosa que en otros sistemas.
  - La modificación de la estructura de datos es más compleja y costosa de realizar.
  - Los tiempos de respuesta se pueden ver afectados cuanto crecen mucho los datos almacenados.

Sistema No Relacional NoSQL

- Ventajas
  - Sistema con gran flexibilidad, ya que permite añadir nuevos campos sin afectar al rendimiento o

a los datos ya existentes. Esto permite tener un esquema de datos flexible con gran capacidad de crecimiento.

- Dispone de una gran versatilidad, ya que utiliza la notación JSON (Javascript Object Notation) para almacenar la información, y por lo que se comunica muy bien con aplicaciones que utilizan estos mismos objetos.
- Es un sistema muy manejable que no necesita mucho mantenimiento ni administración.
- Optimización y alta velocidad de respuesta para consultas cortas.
- Puede comenzar utilizando pocos recursos y escalar según vaya necesitando más.

- Desventajas
  - El sistema no relacional no contiene la característica de la atomicidad, por lo que no siempre contiene datos consistentes en distintas replicas.
  - Al ser de un sistema mucho más nuevo no dispone de estándares claros ni una documentación extensa.
  - Pocas interfaces gráficas para la gestión y diseño.

Una vez analizadas las distintas ventajas e inconvenientes que tiene cada una de las opciones listadas anteriormente, podemos concluir que un almacenamiento de datos no relacional se adapta mejor al sistema. Esta decisión viene condicionada por su gran versatilidad e integración con la notación JSON, utilizada por la aplicación de gestión. Además el sistema de almacenamiento de datos será desplegado en un hardware de baja potencia, por lo que su facilidad de instalación y mantenimiento, sumado a la utilización de bajos recursos hace que sea la opción que mejor se adapta.

Una vez escogido el sistema de almacenamiento no relacional faltaría por escoger la base de datos que se quiere utilizar. Existen distintas bases de datos no relacionales como Redis [32], Cassandra [33] o MongoDB [34]. La utilización de Redis queda descartada por ser una base de datos del tipo clave-valor, la cual no cubriría las necesidades del sistema. Para asegurar la máxima compatibilidad entre el servidor y la base de datos, la utilización de MongoDB sería lo más recomendado, ya que su mayor comunidad y librerías hacen que la implementación sea más estable y robusta que en otras opciones.

Conocidos los requisitos y objetivos relacionados con el modelo de datos, el modelo resultante es el que se muestra en la Figura 6.1, en donde se pueden ver las dos clases resultantes y como están relacionadas entre si de manera 1 a N, esto quiere decir que una pictolista puede contener uno o varios pictogramas.

Una vez decidido que se va a utilizar un sistema de base de datos no relacional, podemos hacer la implementación del modelo de datos final que se va a utilizar en la aplicación, basándonos en el diseño del modelo de datos y el diagrama de clases anteriormente realizado. En este caso al ser un sistema no relacional, la información que se guarda en el sistema de almacenamiento se efectuará mediante objetos JSON, una de las ventajas de MongoDB comentadas anteriormente. Tal y como se puede ver en el Cuadro 6.1, existe un objeto principal asociado a la clase Pictolista, que contiene sus dos atributos como claves (nombre y descripción) y un array de objetos asociados con la clase Pictograma, este objeto pictograma también tiene sus dos atributos como clave (url y time).

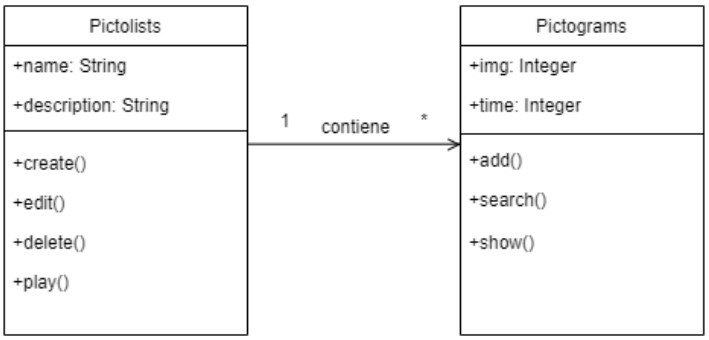

Figura 6.1: Diagrama de Clases

{
***pictolista**: { **name**: 'Lista', **description**: 'Ejemplo de lista',*
***pictograms**: [{ url: '123', time: "60"}] }*
}

6.1: Ejemplo JSON almacenado

6.2. Aplicación de gestión

Para hacer que la aplicación pueda ser utilizada desde múltiples dispositivos y sistemas operativos se utilizará Ionic Framework, un framework MVVM (Model View ViewModel) [35] que utiliza Javascript.

En Ionic Framework es posible utilizar Javascript nativo, ReactJS, Angular, o Vue, aunque para la realización de esta aplicación se ha tomado la decisión de utilizar Angular. Esta elección se ha tomado teniendo en cuenta que Angular es un framework completo que ofrece soluciones para todo el marco de la aplicación, mientras que en otros como ReactJS, es necesario utilizar librerías de terceros las cuales dificultan más las compatibilidades y mantenimientos. Otra de las ventajas de Angular es que ha sido utilizada junto a Ionic desde el principio, por lo que está mucho mejor integrado y existe una mayor comunidad de desarrolladores detrás que con otros frameworks o librerías.

La utilización de Angular para la implementación de la aplicación, permite y facilita la utilización de una adaptación del patrón modelo-vista-controlador MVC, en donde cada una de las funcionalidades y vistas de la aplicación tienen varias partes donde se reparte el trabajo. En Angular una de ellas es la encargada de modelar la vista, mientras la otra, la parte controladora, se encarga de interactuar con la vista y sus elementos. Desde esta parte controladora es también desde donde se llaman a los servicios encargados de realizar peticiones al servidor, que serían el equivalente a la parte modelo del patrón.

Como se ha explicado anteriormente en la Sección 5.3 de la arquitectura del sistema, la aplicación de gestión tiene dos partes claramente diferenciadas y cuyos objetivos finales son diferentes. En primer lugar la aplicación es utilizada para el control del sistema, que sirve para el control de la reproducción de pictogramas, la gestión de pictolistas, y para la configuración de las luces. Por otro lado, la aplicación también tiene un visor de pictogramas, es decir, la parte de la aplicación que se ejecuta en el dispositivo y se encarga únicamente de mostrar los pictogramas que están reproduciéndose. Estas dos partes de la aplicación se podían haber separado en dos aplicaciones distintas, una que se ejecutase en el dispositivo del usuario que controla el sistema, y la otra en el dispositivo que tiene las luces LED y se encarga de mostrar los pictogramas, sin embargo el hecho de juntar las dos partes en la misma aplicación tiene también sus ventajas.

Esta configuración permite que tan solo sea necesario que la aplicación esté instalada en un solo sitio, ya que al ser una aplicación multiplataforma es accesible a través de la red como una aplicación web desde varios dispositivos. Además, esta configuración permite que cualquier dispositivo pueda ser utilizado como visualizador de pictogramas, dando mayor flexibilidad, y haciendo que el desarrollo sea también mucho más eficiente, ya que se pueden aprovechar las clases y servicios encargados de la comunicación, así como la estructura de la misma.

Conexión con ARASAAC

Uno de los requisitos y objetivos principales era el de conseguir que la subida y obtención de los pictogramas fuera lo más sencilla y rápida posible. Una de las primeras posibilidades era la de que fuera el propio usuario el encargado de buscar los pictogramas que quisiera utilizar, y los subiera al servidor mediante la aplicación para su posterior uso en listas. Esta opción permitía que fuera el usuario el que pudiera elegir entre una amplia colección de pictogramas desde distintas fuentes, pero en cambio hacía que el proceso fuese algo lento. Como alternativa, estaba la posibilidad de conectarse a una API que sirviera estos pictogramas, lo que facilitaba en gran medida su búsqueda y utilización, pero limitaba su variedad y las distintas fuentes de donde se podían obtener.

Basándonos en las ventajas y desventajas de cada una de ellas, se decide implementar la conexión con la API de ARASAAC, poniendo por encima del resto su ventaja para la obtención de pictogramas, y necesitando una nueva vista en la aplicación para su búsqueda y selección.

Para realizar esta conexión con la API del Centro Aragonés para la Comunicación Aumentativa y Alternativa es necesario hacer peticiones a dos rutas diferentes, y en las cuales no es necesario autenticarse:

- **https://api.arasaac.org/api/pictograms/es/search/(Búsqueda)**: Es la ruta utilizada para para buscar pictogramas a partir de texto. Cuando se llama a esta ruta, devuelve un array de objetos JSON con las IDs de los pictogramas que corresponden al texto buscado.
- **https://api.arasaac.org/api/pictograms/(ID del pictograma)**: Es la ruta a la cual le pasas la ID de un pictograma y esta te devuelve la imagen que le corresponde. Es la ruta a la cual se llamará cuando sea necesario pintar un pictograma en la aplicación, ya sea cuando se muestre en el visor de pictogramas o en la búsqueda de los mismos.

Para poder hacer uso de la implementación con la API, es necesario crear una vista en la aplicación que permita realizar una búsqueda de pictogramas mediante texto. Para ello se comienza realizando una petición GET a la ruta anteriormente descrita, con el texto introducido enviado como parámetro en la URL. Una vez devuelve la lista de todos los pictogramas en formato JSON, utiliza la segunda ruta para obtener la imagen y dejar al usuario seleccionar aquellos que desea añadir a su lista.

Siguiendo el prototipo de la vista diseñado anteriormente, la implementación final de la vista quedaría tal y como se muestra a continuación en la Figura 6.2, en donde se puede apreciar como se ha realizado una búsqueda de pictogramas con la palabra clave Pájaro, la cual devuelve tres resultados y en donde se han seleccionado dos de ellos.

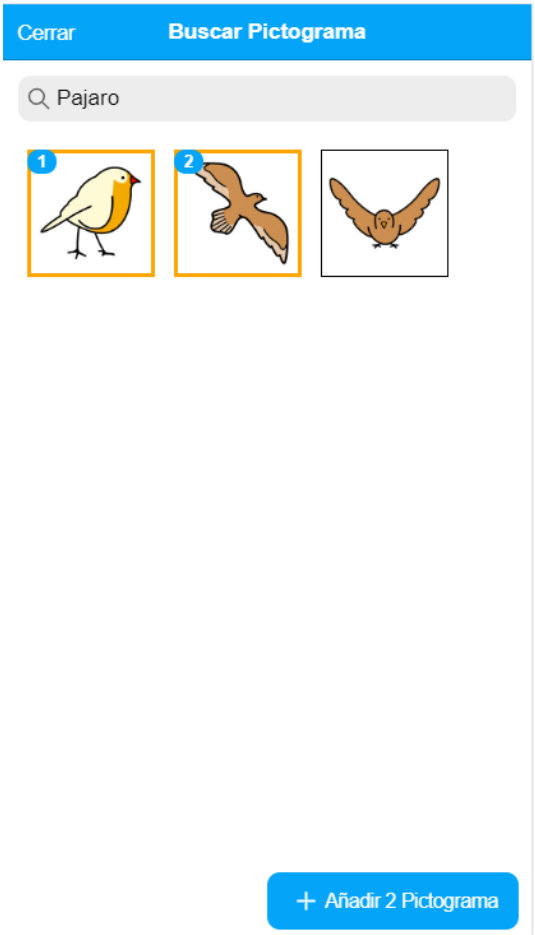

Figura 6.2: Pantalla Búsqueda de Pictogramas

Reproducción de Listas

El visor de pictogramas es la parte encargada de mostrar el pictograma que se está reproduciendo en cada momento. Como es la parte visible de la reproducción, es el encargado de estar emitiendo, con un intervalo de un segundo, el estado de la reproducción al resto de dispositivos mediante el uso de un socket. Eso permite que todos los dispositivos conectados estén perfectamente sincronizados con la reproducción, ya que los datos que se muestran en todos ellos corresponden a los que el propio visor está reproduciendo. Esta emisión continuada en el socket también es utilizada por el resto de la aplicación, ya que deben saber si hay un visor de pictogramas activo con el objetivo de bloquear o desbloquear las funciones de reproducción.

Por otra parte, el visor también debe saber cuando se realizan acciones sobre la reproducción como puede ser pausar, avanzar o modificar el tiempo de reproducción de un pictograma, ya que es el encargado que responder a estas acciones modificando el estado de la reproducción, ya sea avanzando el pictograma que se muestra o pausando el tiempo. Para ello, debe estar escuchando nuevas llamadas al socket con la llegada de datos en varias rutas, con el fin de que no se pierda ninguna acción que se realice sobre el sistema.

Por ejemplo, cuando un usuario elige la pictolista que quiere reproducir, se emite en la ruta pictolist el identificador de la pictolista que ha seleccionado. El visor, que está escuchando en esta ruta del socket, recibe el identificador y realiza una consulta al servidor para que le devuelva los datos de esa lista. Una vez el visor ya conoce los pictogramas que la forman, el tiempo que estará cada uno de ellos, y el orden de reproducción, ya es capaz de reaccionar a las acciones que realice el usuario desde el panel de control sobre esa lista.

Otro de los casos en los que el visor de pictogramas debe de actuar, es cuando es necesario enviar nueva información al servidor para provocar un cambio en el comportamiento de las luces LED. Como el visor de pictogramas es el encargado del control de la reproducción y sabe en todo momento el estado de la misma, también es el responsable de comunicar al servidor estas mismas acciones sobre las luces, por lo que le envía también mediante un socket toda la información necesaria al servidor para el control de las mismas.

Control de la reproducción

La parte del Control de reproducción es la encargada de permitir realizar acciones de control sobre la reproducción que se está realizando en el sistema. El funcionamiento de esta parte se basa en la emisión de las acciones que realiza el usuario por el socket, lo que permite que le lleguen tanto al resto de dispositivos para que tengan la información de las últimas acciones realizadas en el sistema y puedan adaptar sus interfaces, como al visor de pictogramas, ya que como se indicó anteriormente es el que debe modificar la reproducción.

Aunque su trabajo principal se basa en emitir, también es una parte que debe de estar sincronizada con la reproducción, por lo que estará escuchando en el socket en el que emite el visor de pictogramas los datos importantes del estado de la misma. Esta actualización de datos constante que recibe, permite mostrar en tiempo real el estado de la reproducción, el tiempo que lleva de reproducción el pictograma e incluso la imagen del pictograma que se está reproduciendo es ese momento. Además en el momento que deja de recibir esta información por parte del visor de pictoramas, sabe que no hay ninguno conectado y por tanto, debe deshabilitar al usuario la posibilidad de interacción con los controles.

Gestión de listas

Es la parte de la aplicación que permite al usuario gestionar sus listas, creándolas, modificándolas o eliminándolas. Para realizar esta gestión, la aplicación debe de realizar muchas consultas al servidor, ya que es el encargado de toda la interacción con el sistema de almacenamiento de datos. Todas las listas que se creen se enviarán al servidor para que puedan ser almacenadas en la base de datos, y por tanto, persistan entre usos del dispositivo. Por consiguiente, es la parte de la aplicación en la que destaca el uso de peticiones GET, POST, PUT y DELETE para realizar el intercambio de información con la parte servidor de la aplicación y con la API de ARASAAC.

Desde esta parte también será necesario hacer envíos de datos por el socket de la aplicación, ya que desde aquí se debe seleccionar la lista que se quiere comenzar a reproducir. Una vez seleccionada, pulsando en el botón de reproducir, el identificador de esta lista será enviado por el socket hasta el visor de pictogramas, por lo que también es importante que esta parte esté pendiente de cambios para saber si hay algún visor activo en ese momento y permitir así utilizar el botón de reproducción.

6.3. API

La API del sistema es la parte central del software, ya que se encarga de proporcionar un medio de comunicación entre distintos dispositivos y módulos de la aplicación, de realizar las comunicaciones con la base de datos, y de ejecutar el script encargado del control de las luces LED.

Para la implementación de esta API se ha utilizado NodeJS, la cual utiliza MongoDB como sistema de almacenamiento. Además, para permitir la comunicación en tiempo real entre distintos dispositivos se utilizan WebSockets.

Las funcionalidades de las que se encarga el servidor pueden ser separadas en tres grupos distintos:

- **Gestión de Pictolistas**: El servidor es la parte del sistema que interactúa con la base de datos, por lo que es el encargado de gestionar la listas que crean, editan o eliminan los usuarios. Esto significa que recibe las peticiones que se realizan desde la aplicación e interacciona con el sistema de almacenamiento de datos para lograr la

persistencia de la información entre sesiones.

- **Sincronización de dispositivos**: Es la parte encargada de recibir información de los distintos dispositivos que interactúan con el sistema y de difundirla al resto de dispositivos conectados, lo que permite que exista sincronía en la información de la que disponen todos ellos en tiempo real.

- **Control de las luces LED**: Aunque el servidor no es el encargado directo del control de las luces LED es el que recibe la información de las acciones que realiza el usuario, y por tanto, es el encargado de la ejecución pasando los parámetros necesarios al script de las luces.

Gestión de Pictolistas

Como se mencionaba anteriormente, la parte servidor es la encargada de interactuar con el sistema de almacenamiento de datos, y por tanto, todo lo que tenga que ver con la gestión de pictolistas debe pasar por el servidor para que esta información se guarde. Todos estos datos son enviados desde la aplicación al crear, editar o eliminar una lista, y por tanto el servidor debe proporcionar servicios en donde pueda recibir esta información. Estas rutas están en formato API Rest, las cuales hacen uso de los distintos métodos de petición que ofrece HTTP para cada una de las acciones (GET, POST, PUT, DELETE) [36]. En el código del apartado 6.3 se puede apreciar la creación de todas las rutas HTTP en las que responde el servidor, así como las llamadas a las funciones encargadas de la respuesta para cada uno de ellos.

Una vez se realizan estas peticiones, el servidor interactúa con la base de datos MongoDB para modificar los datos guardados y devuelve una respuesta a la aplicación para que actualice la interfaz.

Sincronización de dispositivos

Uno de los puntos importantes era el de conseguir que varios dispositivos pudieran interaccionar al mismo tiempo con el sistema. Para conseguir esto es necesario que exista un intercambio de información en tiempo real entre todos los dispositivos, para lo que se han utilizado websockets, en concreto la librería Socket.io [37].

La utilización de Websockets permite que tanto el servidor como las aplicaciones conectadas no tengan que estar haciendo peticiones HTTP constantemente para conocer el estado del sistema. Hacer uso de Websockets permite realizar comunicaciones TCP bidireccionales basadas en eventos, por lo que la información se actualiza en tiempo real [38]. Esta característica es de vital importancia para el sistema, ya que es necesario que la información esté sincronizada en todo momento y la información que se muestre en todos los sitios sea la misma. Esta sincronización es necesaria entre las aplicaciones de gestión que manejan un dispositivo, entre la aplicación de gestión y el visor de pictogramas, y ambas aplicaciones con el servidor, ya que es el encargado de manejar las luces LED.

Código 6.1: En esta figura se presenta el código correspondiente al fichero de rutas en donde se definen endpoints de la API.

```
1  var express = require('express');
2  var router = express.Router();
3
4  // Require the controllers
5  const pictolist_controller = require('../controllers/pictolist.controller');
6
7  // GET Pictolists
8  router.get('/pictolists', pictolist_controller.getPictolists);
9  router.get('/pictolists/:id', pictolist_controller.getPictolist);
10
11 // POST Pictolists
12 router.post('/pictolists', pictolist_controller.newPictolist);
13
14 // PUT Pictolists
15 router.put('/pictolists/:id', pictolist_controller.editPictolist);
16
17 // DELETE Pictolists
18 router.delete('/pictolists/:id', pictolist_controller.deletePictolist);
19
20
21 module.exports = router;
```

En el código del apartado 6.3 pueden verse todas las rutas de los sockets que se han creado y en donde escucha el servidor, así como sus respectivas respuestas para cada una de las llamadas que se realizan. Por ejemplo, como puede verse en la línea 26 del código, cuando recibe una llamada al endpoint LEDS ejecuta una función encargada del control de las luces LED.

Control de luces LED

Para el control de las luces LED ha sido necesario utilizar Python como lenguaje principal. Esto es debido a que la conexión directa de las luces LED desde NodeJS era muy compleja de integrar en el propio servidor, ya que la mayoría de librerías están deprecadas y se suelen utilizar lenguajes como C, C++, o lenguajes de script como Python, en donde su integración está desarrollada y mantenida por los propios fabricantes de los dispositivos. Por tanto, existe un script en Python encargado del control de las luces LED que actúa dependiendo de los argumentos iniciales con los que se le invoque, y es el propio servidor NodeJS el que lo ejecuta mediante subprocesos (child process). Esta llamada al script como subproceso se puede ver en la línea 9 del código del apartado 6.3, en donde primero se comprueba que no exista ningún script en ejecución y se ejecuta uno nuevo con los argumentos necesarios.

Código 6.2: En esta figura se presenta el código correspondiente al fichero donde se definen las rutas de los sockets que maneja la API.

```
let io = require('socket.io')(server);

io.on('connection', (socket) => {

  socket.on('actions', (actions) => {
    io.emit('actions', {action: actions});
  });

  socket.on('pictolist', (pictolist) => {
    io.emit('pictolist', pictolist);
  });

  socket.on('status', (obj) => {
    io.emit('status', obj);
  });

  socket.on('changeTime', (obj) => {
    io.emit('changeTime', obj);
  });

  socket.on('time', (time) => {
    io.emit('time', {time: time});
  });

  socket.on('leds', (leds) => {
    pythonScripts(leds);
  });
});
```

Código 6.3: En esta figura se presenta el código correspondiente al la parte del servidor encargado de ejecutar la llamada al script de las luces.

```
function pythonScripts(leds) {
  if(ls) {
    ls.stdin.pause();
    ls.kill();
  }

  if(leds.status != "paused") {
    const spawn = require('child_process').spawn;
    ls = spawn('python', ['./py/leds.py', leds.currentTime, leds.totalTime, leds.mode]);

    ls.stdout.on('data', (data) => {
      console.log(`stdout: data‘); ); ls.stderr.on('data', (data) => console.log(‘stderr :{data}`);
    });

    ls.on('close', (code) => {
      console.log(`Leds script execution ended`);
    });
  }
}
```

Script LEDs

Como se dijo en el apartado anterior, las luces LED se controlan mediante un script en Python, ya que el fabricante de las luces LED tiene una librería específica para este lenguaje y por tanto facilita mucho las labores de implementación, en este caso utiliza la de neopixels.

Al ser un fichero script con nula comunicación con el sistema y necesitar algunos datos para la personalización de los LEDs, espera que junto a su ejecución lleguen ciertos argumentos de entrada, los cuales servirán para personalizar ciertos aspectos de la ejecución como el patrón que siguen los LEDs o el tiempo que emplean. En el Anexo F se puede ver el código Python que interactúa directamente con las luces LED.

6.4. Raspberry Pi

La Raspberry Pi es la pieza hardware que contiene todo el software necesario para hacer funcionar el sistema y proporciona el enlace físico con las luces LED. En la Figura 6.3 puede verse un ejemplo de como es una Raspberry Pi 4.

Existen múltiples sistemas operativos y distribuciones compatibles con la Raspberry Pi, a continuación se detallarán los más importantes y se explicará el porqué de la selección escogida.

Raspberry Pi OS [39]: Es un sistema operativo basado en la distribución Debian de Linux.

Una de sus gran ventajas es que está muy bien optimizado para el hardware de la Raspberry Pi y además proporciona la interfaz de usuario necesaria para mostrar los pictogramas.

Ubuntu Core [40]: Es la versión Internet de las Cosas (Internet of things, IoT) del sistema operativo Ubuntu, basado también en la distribución Debian de Linux. Su gran ventaja es la posibilidad de utilizar Snap para la instalación de paquetes software.

Windows 10 IoT Core [41]: Es una versión de Windows 10 optimizada para ser utilizada en dispositivos pequeños sin pantalla y que se ejecutan en dispositivos ARM y x86 o x64.

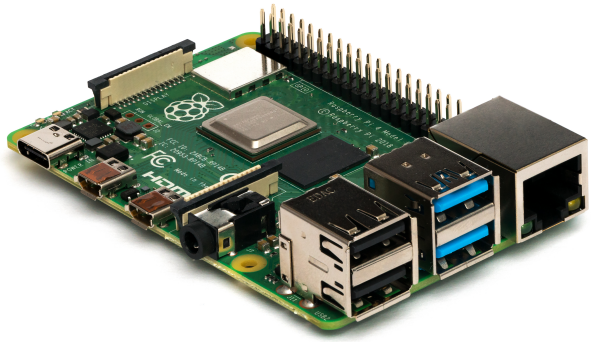

Figura 6.3: Ejemplo de una Raspberry Pi 4

Para el uso que se necesita dar al sistema actual, Raspberry Pi OS es la mejor opción. Esto es debido a que es un sistema operativo mucho más maduro y más preparado para funcionar con la Raspberry Pi, ya que otros como Ubuntu Core son más recientes y contienen más errores. Además Raspberry Pi OS ya esta preparado para utilizar herramientas como Python, necesario para el proyecto.

Una vez instalado y configurado el sistema operativo en la tarjeta SD la Raspberry Pi, se debe continuar con la instalación de las herramientas necesarias para el funcionamiento del sistema. Para la ejecución del servidor API es necesario instalar **NodeJS** junto a **MongoDB** como sistema de almacenamiento. Para crear nuevas compilaciones de la aplicación de gestión es necesario tener instalado **Ionic Framework** junto con **Angular**, pero tan solo será necesario instalar un servidor web como Apache [29] o **NGINX** [42] para mostrar la aplicación como un servicio web. Es posible encontrar un manual de instalación en el Anexo A.2 de este mismo documento.

6.5. Prototipo

Tras la búsqueda del mejor hardware a implementar en el proyecto, se han decidido utilizar para el prototipo final aquellos dispositivos que mejor se han adaptado al sistema y de los cuales se han hablado en la fase de diseño. Las características del hardware utilizado se pueden ver en las Tablas 6.1, 6.2 y 6.3. El uso de este hardware no es del todo restrictivo, ya que será posible encontrar otro

hardware compatible si comparte características similares a las que se detallan a continuación:

Raspberry Pi 4 [43]

Procesador	ARM Cortex-172 con cuatro núcleos a 1,5 GHz
GPU	VideoCore VI (con soporte para OpenGL ES 3.x)
Memoria	4GB LPDDR4 SDRAM
Conectividad	Bluetooth 5.0, Wi-Fi 802.11ac, Gigabit Ethernet
Puertos	GPIO 40 pines 2 x micro HDMI 2 x USB 2.0 2 x USB 3.0 CSI (cámara Raspberry Pi) DSI (pantalla tácil)

Tabla 6.1: Características Raspberry Pi 4

Waveshare 4inch HDMI LCD [44]

Resolución	800 x 480
Táctil	Si (Aunque no es necesario)
Tecnología	IPS
Compatibiliad	Compatible y conexión directa con cualquier revisión de Raspberry Pi (excepto Pi 1 modelo B o Pi Zero, que requiere un cable HDMI)
Conexiones	HDMI

Tabla 6.2: Características Waveshare 4inch HDMI LCD

Tira de LED Adafruit [28]

Número de LEDs	30 (No se utilizan todas)
RGB	Si
Medidas	0.5m
Conexiones	Cable JST PH de 3 conectores
Extendible	Si
Acortable	No

Tabla 6.3: Características tira de LED Adafruit

Para la implementación del prototipo ha sido necesario realizar ciertas conexiones entre la Raspberry Pi 4 con las tiras de luces LED. El esquema de conexiones que se ha seguido es el que se puede ver en la Figura 6.4, extraído directamente de la web del fabricante [28].

Debido a que la Raspberry Pi no tiene potencia suficiente para alimentar las LEDs, es necesario el uso de un alimentador externo, el cual servirá para alimentar tanto a la tira de luces LED como a la Raspberry Pi. Esto provoca que sea necesario el uso de un diodo 1N4001 el cual asegura una entrada de 5v para la alimentación de los LEDs.

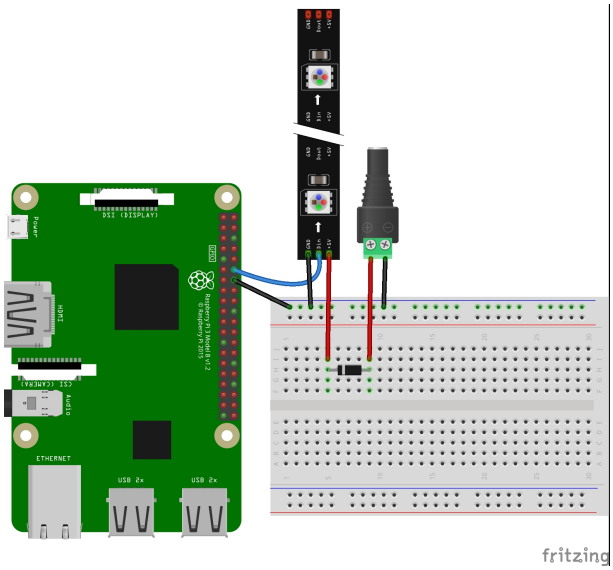

Figura 6.4: Esquema de conexiones de la Raspberry Pi

Para realizar las conexiones que se muestran en el esquema de la Figura 6.4, será necesario conectar el pin GND de la tira de luces LED y el GPIO GND de la Rasbperry Pi, al polo negativo de la fuente de alimentación. En cuanto al polo positivo de la fuente de alimentación, este debe ir conectado al conector 5v de la tira de luces LED y a la Raspberry Pi, con la particularidad de que la conexión entre la fuente de alimentación y las luces, debe tener en el medio un diodo con la raya apuntando a las luces. El último paso para acabar con la conexión de las luces LED sería la conexión encargada de enviar las señales de control, para el cual se tendrá que conectar la entrada DIN de la tira de luces al GPIO18 de la Raspberry Pi. Es importante que esta conexión se realice al GPIO18, ya que en la Raspberry Pi es un pin PCM_CLK y envia señales de reloj que permiten controlar todos los leds de la tira con una única salida.

El prototipo general de la implementación del cableado de las luces LED a la Raspberry Pi puede verse a continuación en las Figuras 6.5, 6.6 y 6.7, donde se sigue el esquema de conexiones explicado anteriormente.

Una vez conectadas la tira de luces LED a las Raspberry Pi, faltarían las conexiones de la pantalla LCD al dispositivo para tener todas las conexiones necesarias completas. La pantalla LCD va conectada encima de la Raspberry Pi mediante los pines GPIO de la misma, por lo que tendrá que ser colocada justo encima de los pines utilizados para las conexiones de las luces LED, tal y como se ve la conexión en la Figura 6.8, en donde la pantalla se conecta de manera que se coloca encima de los primeros pines, y por debajo se mantienen las conexiones de las luces LED realizadas con anterioridad. Por último, mediante el adaptador incluido en la pantalla se conecta la entrada HDMI de la misma a la micro HDMI de la Raspberry Pi.

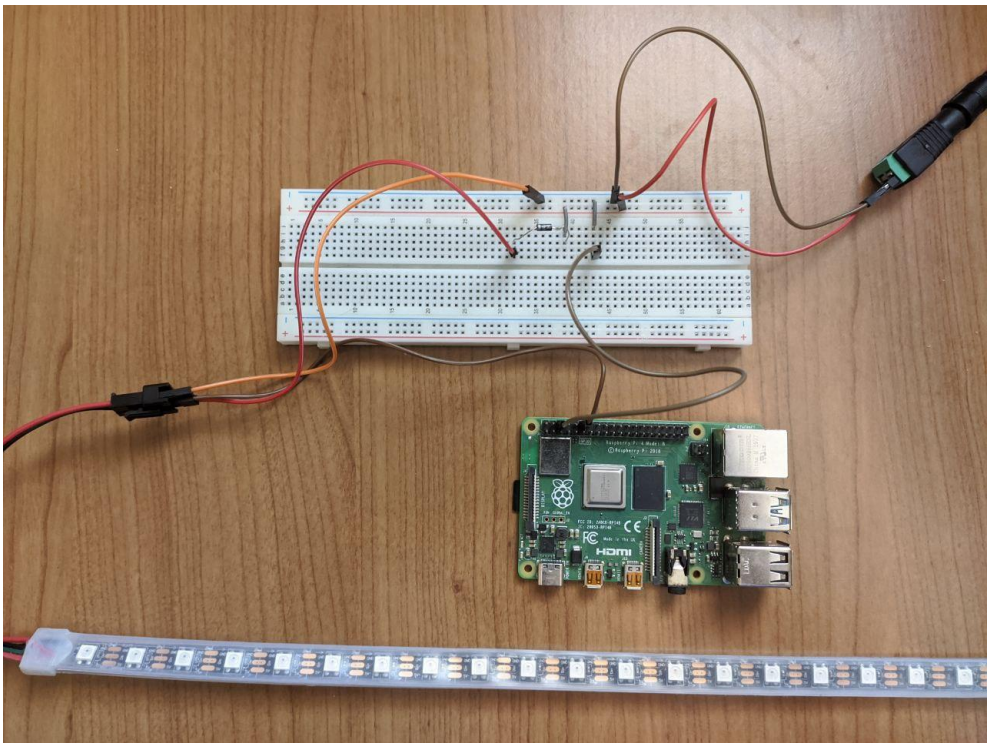

Figura 6.5: Prototipo del cableado de las luces LED

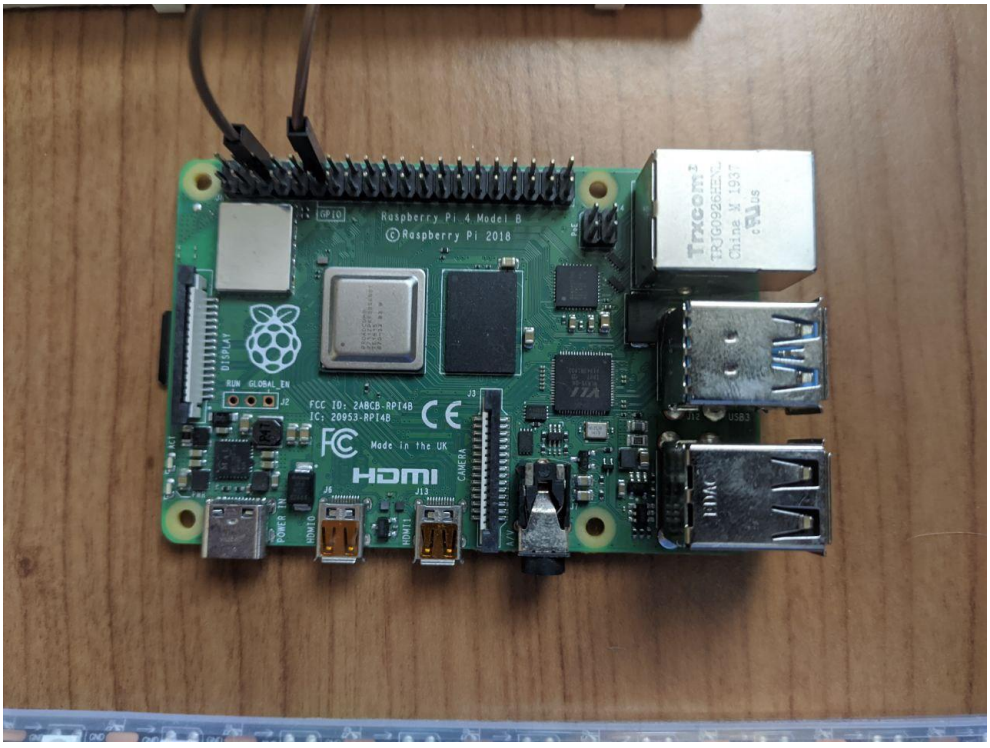

Figura 6.6: Conexiones del sistema en la Raspberry Pi

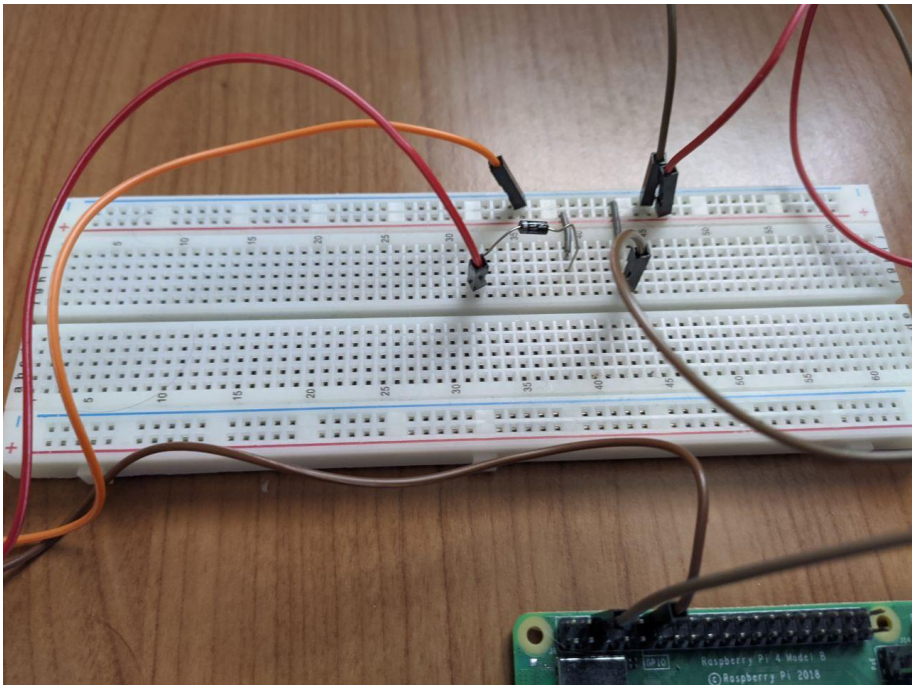

Figura 6.7: Conexiones del sistema en la Protoboard

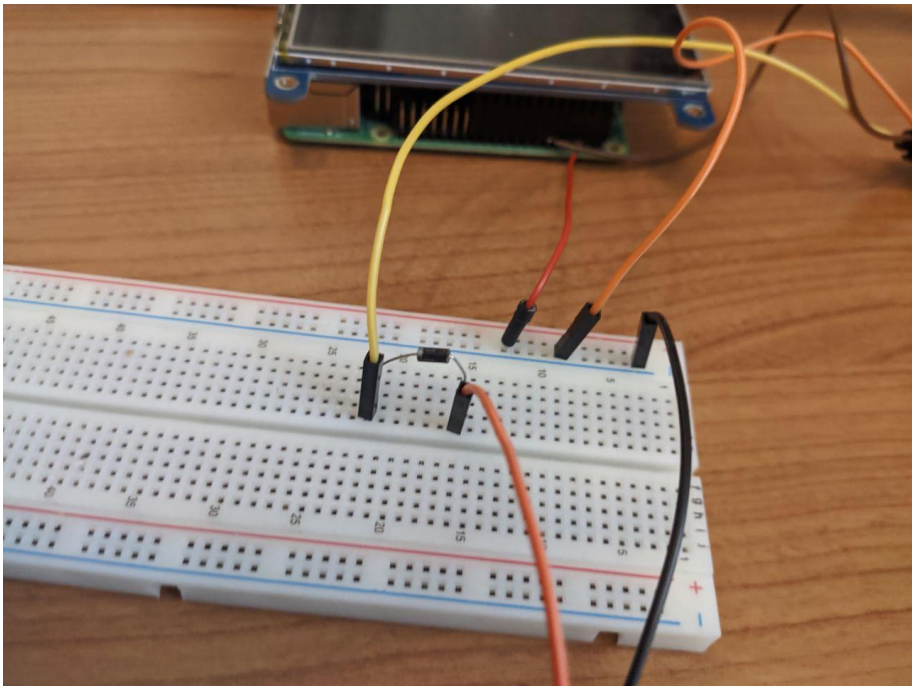

Figura 6.8: Conexiones de la pantalla

Una vez estén conectadas las pantallas y las luces LED, tal y como se ha indicado anteriormente, el cableado del sistema ya estaría acabado y el dispositivo listo para funcionar. En la Figura 6.9 se puede apreciar el cableado final en donde están conectados simultáneamente las luces LED y la pantalla LCD.

Tras realizar estas conexiones el sistema ya cuenta con una pantalla y una tira de luces LED funcionales, por lo que el prototipo estaría preparado. A partir de aquí el sistema puede ser portado a algún dispositivo contenedor para realizar su ensamblado, aunque sería recomendable cablear el sistema de forma directa entre sus componentes, ya que esto reduce todavía más el espacio que ocupa, y lo hace más resistente a desconexiones accidentales.

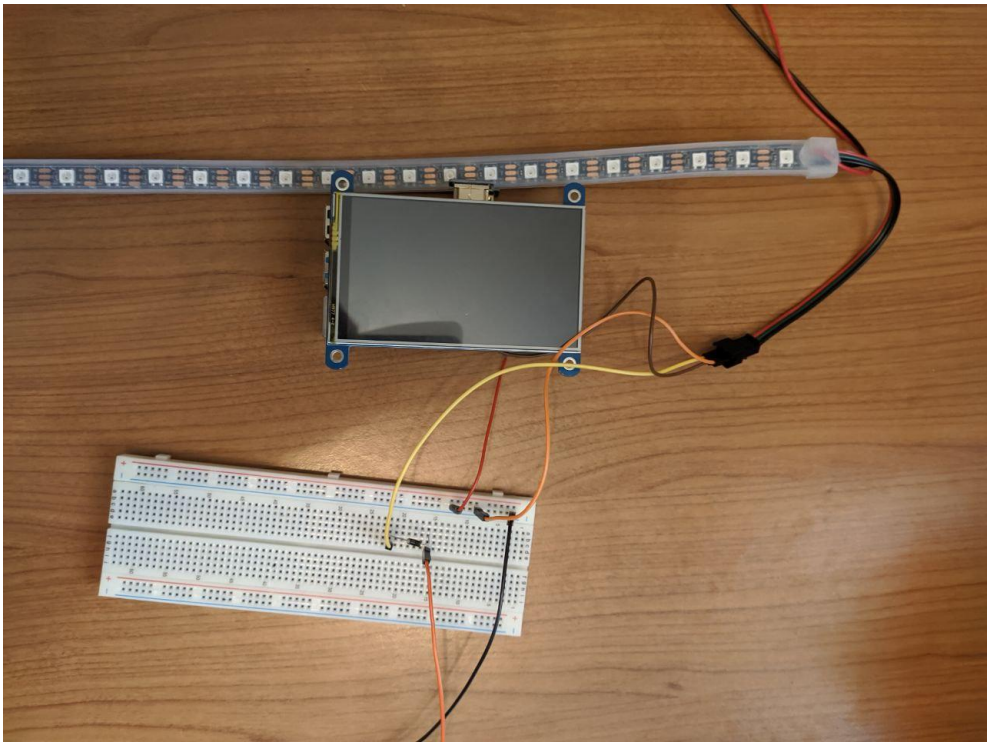

Figura 6.9: Conexiones finales

7

Pruebas

En este capítulo se realizarán pruebas al sistema que permitan asegurar que su funcionamiento es el correcto y esperado. Para esto, se realizarán dos tipos de pruebas, las de caja blanca y las de caja negra.

Las pruebas de caja blanca se centrarán en que los flujos de ejecución y los valores que se devuelven sean los esperados, mientras que en las pruebas de caja negra se comprobarán que las salidas que proporciona el sistema son adecuadas a las entradas enviadas por el usuario [45].

7.1. Pruebas de Caja Blanca

A continuación se realizarán las pruebas de caja blanca sobre el funcionamiento total del sistema.

Caso de Prueba	Acceso a la aplicación
Criterios de éxito	El usuario accede correctamente a la aplicación
Criterios de error	El usuario no puede acceder a la aplicación
Precondiciones	El usuario debe estar conectado en la misma red que el dispositivo y este debe estar ejecutándose
Resultado	El usuario accede correctamente a la aplicación y se encuentra en el menú principal

Tabla 7.1: Prueba de acceso a la aplicación

Caso de Prueba	Navegación entre menús
Criterios de éxito	El usuario puede moverse correctamente entre los menús a la aplicación
Criterios de error	El usuario no puede moverse entre menús
Precondiciones	El usuario debe tener acceso a la aplicación
Resultado	El usuario se mueve correctamente entre menús

Tabla 7.2: Prueba de navegación entre menús

Caso de Prueba	Acceso a la creación de pictolista
Criterios de éxito	El usuario accede a la vista de creación de listas
Criterios de error	El usuario no puede acceder a la vista
Precondiciones	El usuario debe tener acceso a la aplicación
Resultado	El usuario accede correctamente a la vista de creación de pictolistas

Tabla 7.3: Prueba de acceso a la creación de pictolista

Caso de Prueba	Creación de Pictolista
Criterios de éxito	El usuario puede crear correctamente la pictolista
Criterios de error	El usuario no puede crear una nueva lista
Precondiciones	El usuario debe tener acceso a la aplicación y conexión con el servidor
Resultado	El usuario puede crear correctamente una lista después de introducir los datos

Tabla 7.4: Prueba de creación de pictolista

Caso de Prueba	Acceso a añadir pictogramas
Criterios de éxito	El usuario puede acceder correctamente al modal de añadir pictogramas
Criterios de error	El usuario no puede acceder al modal de añadir pictogramas
Precondiciones	El usuario debe tener acceso a la aplicación
Resultado	El usuario puede acceder correctamente al modal de añadir pictogramas

Tabla 7.5: Prueba de acceso a añadir pictogramas

Caso de Prueba	Búsqueda de pictogramas
Criterios de éxito	El usuario puede buscar pictogramas a través del texto introducido
Criterios de error	El usuario no puede buscar pictogramas
Precondiciones	El usuario debe tener acceso a la aplicación y la API de ARASAAC debe estar disponible
Resultado	El usuario puede buscar pictogramas a través de texto correctamente

Tabla 7.6: Prueba de búsqueda de pictogramas

Caso de Prueba	Selección de pictogramas
Criterios de éxito	El usuario puede seleccionar los pictogramas que quiere añadir
Criterios de error	El usuario no puede seleccionar los pictogramas
Precondiciones	El usuario debe tener acceso a la aplicación y la API de ARASAAC debe estar disponible
Resultado	El usuario puede seleccionar los pictogramas que desea añadir

Tabla 7.7: Prueba de selección de pictogramas

Caso de Prueba	Añadir pictogramas
Criterios de éxito	El usuario puede añadir pictogramas a una lista
Criterios de error	El usuario no puede añadir pictogramas
Precondiciones	El usuario debe tener acceso a la aplicación y la API de ARASAAC debe estar disponible
Resultado	El usuario puede añadir los pictogramas a la lista correctamente

Tabla 7.8: Prueba de añadir pictogramas

Caso de Prueba	Eliminar pictogramas
Criterios de éxito	El usuario puede eliminar pictogramas añadidos a una lista
Criterios de error	El usuario no puede eliminar pictogramas en una lista
Precondiciones	El usuario debe tener acceso a la aplicación
Resultado	El usuario puede eliminar los pictogramas que están en una lista correctamente

Tabla 7.9: Prueba de eliminar pictogramas

Caso de Prueba	Modificar tiempo pictogramas
Criterios de éxito	El usuario puede modificar el tiempo de cada pictograma añadido a una lista
Criterios de error	El usuario no puede modificar el tiempo de los pictogramas
Precondiciones	El usuario debe tener acceso a la aplicación
Resultado	El usuario puede modificar el tiempo de los pictogramas correctamente

Tabla 7.10: Prueba de modificar tiempo pictogramas

Caso de Prueba	Modificar orden pictogramas
Criterios de éxito	El usuario puede modificar el orden en el que se reproduce cada pictograma
Criterios de error	El usuario no puede modificar el orden de los pictogramas
Precondiciones	El usuario debe tener acceso a la aplicación
Resultado	El usuario puede modificar el orden de reproducción de los pictogramas correctamente

Tabla 7.11: Prueba de modificar orden pictogramas

Caso de Prueba	Crear pictolista
Criterios de éxito	El usuario puede crear una pictolista correctamente
Criterios de error	El usuario no puede crear nuevas pictolistas
Precondiciones	El usuario debe tener acceso a la aplicación y conexión con el servidor
Resultado	El usuario puede crear listas correctamente

Tabla 7.12: Prueba de creación de pictolistas

Caso de Prueba	Editar pictolista
Criterios de éxito	El usuario puede modificar una pictolista correctamente
Criterios de error	El usuario no puede modificar pictolistas
Precondiciones	El usuario debe tener acceso a la aplicación y conexión con el servidor
Resultado	El usuario puede editar las listas creadas correctamente

Tabla 7.13: Prueba de edición de pictolistas

Caso de Prueba	Eliminar pictolistas
Criterios de éxito	El usuario puede eliminar una pictolista correctamente
Criterios de error	El usuario no puede eliminar pictolistas
Precondiciones	El usuario debe tener acceso a la aplicación y conexión con el servidor
Resultado	El usuario puede eliminar las listas creadas correctamente

Tabla 7.14: Prueba de eliminar pictolistas

Caso de Prueba	Reproducir pictolistas
Criterios de éxito	El usuario puede reproducir na pictolista y establecerla como activa correctamente
Criterios de error	El usuario no puede reproducir pictolistas
Precondiciones	El usuario debe tener acceso a la aplicación, conexión con el servidor y tener un visor de pictogramas activo
Resultado	El usuario puede reproducir las listas correctamente

Tabla 7.15: Prueba de reproducir pictolistas

Caso de Prueba	Detener reproducción de pictolistas
Criterios de éxito	El usuario puede detener la reproducción de una pictolista
Criterios de error	El usuario no puede detener la reproducción de una pictolista
Precondiciones	El usuario debe tener acceso a la aplicación, conexión con el servidor y tener un visor de pictogramas activo
Resultado	El usuario puede detener la reproducción correctamente

Tabla 7.16: Prueba de detener reproducción pictolistas

Caso de Prueba	Reproducción de pictogramas
Criterios de éxito	El usuario puede reproducir un pictograma
Criterios de error	El usuario no puede reproducir un pictograma
Precondiciones	El usuario debe tener acceso a la aplicación, conexión con el servidor y tener un visor de pictogramas activo
Resultado	El usuario puede reproducir un pictograma correctamente, este se muestra en el visor y hace que se avance el tiempo de reproducción

Tabla 7.17: Prueba de reproducción de pictogramas

Caso de Prueba	Pausar reproducción de pictogramas
Criterios de éxito	El usuario puede pausar la reproducción de un pictograma
Criterios de error	El usuario no puede pausar la reproducción
Precondiciones	El usuario debe tener acceso a la aplicación, conexión con el servidor y tener un visor de pictogramas activo
Resultado	El usuario puede pausar la reproducción de un pictograma correctamente, este se muestra en el visor y hace que se pause el tiempo de reproducción

Tabla 7.18: Prueba de pausar reproducción de pictogramas

Caso de Prueba	Avanzar reproducción de pictogramas
Criterios de éxito	El usuario puede avanzar la reproducción al siguiente pictograma
Criterios de error	El usuario no puede avanzar la reproducción de pictogramas
Precondiciones	El usuario debe tener acceso a la aplicación, conexión con el servidor y tener un visor de pictogramas activo
Resultado	El usuario puede avanzar la reproducción de un pictograma correctamente y el siguiente pictograma en la lista se vuelve el activo

Tabla 7.19: Prueba de avanzar reproducción de pictogramas

Caso de Prueba	Retroceder reproducción de pictogramas
Criterios de éxito	El usuario puede retroceder la reproducción al anterior pictograma
Criterios de error	El usuario no puede retroceder la reproducción de pictogramas
Precondiciones	El usuario debe tener acceso a la aplicación, conexión con el servidor y tener un visor de pictogramas activo
Resultado	El usuario puede retroceder la reproducción de un pictograma correctamente y el anterior pictograma en la lista se vuelve el activo

Tabla 7.20: Prueba de retroceder reproducción de pictogramas

Caso de Prueba	Modificar tiempo de reproducción
Criterios de éxito	El usuario puede modificar el tiempo de reproducción del pictograma activo
Criterios de error	El usuario no puede modificar el tiempo de la reproducción de pictogramas
Precondiciones	El usuario debe tener acceso a la aplicación, conexión con el servidor y tener un visor de pictogramas activo
Resultado	El usuario puede modificar el tiempo de la reproducción de un pictograma activo correctamente

Tabla 7.21: Prueba de modificar tiempo de reproducción

Caso de Prueba	Configuración LEDs
Criterios de éxito	El usuario puede configurar y personalizar el comportamiento de las LEDs y se guarda su configuración correctamente
Criterios de error	El usuario no puede configurar el uso de las LEDs
Precondiciones	El usuario debe tener acceso a la aplicación y conexión con el servidor
Resultado	El usuario puede configurar las LEDs y su configuración se guarda correctamente

Tabla 7.22: Prueba de configuración LEDs

Caso de Prueba	Visor de pictogramas
Criterios de éxito	El visor de pictogramas responde correctamente al reproductor y muestra los pictogramas que corresponden correctamente
Criterios de error	El visor no muestra los pictogramas adecuados
Precondiciones	El usuario debe tener acceso a la aplicación y conexión con el servidor
Resultado	El visor de pictogramas muestra las imágenes y responde a la aplicación correctamente

Tabla 7.23: Prueba de visor de pictogramas

Caso de Prueba	Visor desconectado
Criterios de éxito	El usuario no puede reproducir pictolistas ni utilizar el reproductor de la aplicación
Criterios de error	El usuario puede reproducir listas o utilizar el reproductor
Precondiciones	El usuario debe tener acceso a la aplicación y el visor no estar activo
Resultado	El reproductor está bloqueado y no se permite reproducir nuevas listas

Tabla 7.24: Prueba de visor desconectado

7.2. Pruebas de Caja Negra

Una vez realizadas las pruebas de caja blanca se comenzarán las pruebas de caja negra, las cuales sirven para comprobar que la salida que produce el sistema a una entrada concreta es la correcta y esperada. En este caso las pruebas de caja negra se centrarán en la introducción de datos de una pictolista, y como estos responden en el sistema.

Datos Introducidos	**Resultado Esperado**	**Resultado Obtenido**
'Actividades Tarde'	Nombre introducido válido	Nombre aceptado
' '	Introducir un nombre es obligatorio	Nombre obligatorio
'Actividades Tarde' (Se permiten duplicados)	Nombre introducido válido	Nombre introducido válido
'Introducción de un nombre muy largo para ver como se comporta'	Nombre introducido válido	Nombre aceptado

Tabla 7.25: Prueba de introducir nombre

Datos Introducidos	**Resultado Esperado**	**Resultado Obtenido**
'Actividades que se realizan por la tarde'	Descripción válida	Descripción aceptada
' '	Descripción vacía válida	Descripción aceptada
'Introducción de una descripción de muchos caracteres para ver como se comporta'	Descripción introducida válida	Descripción introducida aceptada

Tabla 7.26: Prueba de introducir descripción

Datos Introducidos	**Resultado Esperado**	**Resultado Obtenido**
Sin introducir pictogramas	Al menos un pictograma necesario	Pictogramas no válidos
Introduciendo uno o más pictogramas	Número de pictogramas válido	Pictogramas aceptados

Tabla 7.27: Prueba de introducir pictogramas

Datos Introducidos	Resultado Esperado	Resultado Obtenido
Pájaro’	Imágenes y pictogramas de pájaros	Tres pictogramas con pájaros
’ ’	No aparecen pictogramas	No hay resultados que mostrar
’Raspberry’	No se encuentran pictogramas	No hay resultados a la búsqueda

Tabla 7.28: Prueba de búsqueda de pictogramas

Datos Introducidos	Resultado Esperado	Resultado Obtenido
0:00	Tiempo de pictograma no puede ser cero	Tiempo no aceptado
0:01	Tiempo de pictograma válido	Tiempo pictograma válido y funcional
59:59	Tiempo válido	Tiempo válido y funcional

Tabla 7.29: Prueba de introducción de tiempo de pictogramas

Datos Introducidos	Resultado Esperado	Resultado Obtenido
Idioma	Cambio de idioma de la aplicación	Idioma cambia
Desactivar luces LED	Luces LED desactivadas	Las luces LED se desactivan
Activar luces LED	Luces LED activadas	Las luces LED se activan
Personalización luces LED	Las luces LED responden a los datos introducidos	Las luces LED responden a los configuración introducida

Tabla 7.30: Prueba de introducir configuración

8

Conclusiones

Una vez finalizadas las fases de análisis, diseño e implementación del sistema, y conociendo más a fondo como ha quedado el dispositivo final, se pueden sacar algunas conclusiones y análisis de la realización de este proyecto a partir de los objetivos iniciales planteados.

El objetivo principal del proyecto es el de diseñar e implementar un sistema capaz de guiar la realización de actividades mediante pictogramas, y de establecer un sistema que indique el tiempo para cada una mediante un código basado en luz y color.

La principal aportación del proyecto es la de la creación de un dispositivo de gran utilidad y ayuda para personas con Trastorno del Espectro Autista, que adapta uno de los medios y herramientas que utilizan normalmente de forma física, como son los pictogramas, a medios digitales y fácilmente accesibles. Una de las partes que más destaca en el proyecto es la representación del tiempo restante de una actividad, representada mediante pictogramas, mediante el uso de luces y colores, parte novedosa del proyecto y que se ha podido implementar de forma satisfactoria. Esta adaptación permite además que los educadores, profesores o padres que desean utilizarlo, puedan hacerlo de forma sencilla y con un fácil acceso a una gran cantidad de pictogramas, por lo que se considera que el objetivo principal del proyecto ha sido cumplido.

Para llevar a cabo el proyecto se hará uso de tecnologías y dispositivos del Internet de las Cosas (Internet of Things, IoT) de bajo coste, ya que la utilización de dispositivos como pueden ser Arduino o Raspberry Pi permiten que el coste del proyecto sea muy accesible para muchas personas y se faciliten las prácticas de "hazlo tu mismo" (Do it Yourself, DIY en inglés)

Este objetivo se considera cumplido teniendo en cuenta que el coste del hardware utilizado en el prototipo final es muy asequible para un dispositivo de esas características, se utiliza OPEN-HARDWARE como la Raspberry Pi, y se utilizan dispositivos IoT. Todo esto permite que la replica del proyecto por parte de otras personas sea bastante sencilla, lo que permitiría su utilización en varios lugares, como por ejemplo en una clase y en un domicilio.

Montaje de un prototipo hardware que permita estudiar la interacción y viabilidad del sistema que ha sido propuesto.

Finalmente se ha conseguido implementar un sistema completamente funcional, que cumple los objetivos y requisitos planteados inicialmente, y que a pesar de ser un primer prototipo y por tanto su resistencia a golpes es baja, ya se podría comenzar a usar en entornos de prueba reales. Por tanto, aunque este objetivo también se considera cumplido, se esperaban haber podido realizar estas pruebas en entornos reales, lo cual no ha sido posible y se explica a continuación en el apartado de limitaciones.

Conocer en mayor profundidad las necesidades que tienen las personas con TEA, obteniendo información de expertos en la educación y formación de personas con TEA, así como de expertos en desarrollos tecnológicos para el TEA.

Para el cumplimiento de este objetivo han sido de gran utilidad las entrevistas informales realizadas con expertos y profesionales que trabajan en la educación para personas con TEA. Estas entrevistas informales, realizadas con la directora del centro de ayuda Alenta [23], han sido de gran utilidad, ya que fueron utilizadas junto a la lectura de estudios y publicaciones, para obtener los requisitos del sistema, poder saber como trabajan, y conocer las necesidades tanto de los educadores, que utilizarían la aplicación, como de las personas con TEA que utilizarían el sistema. Por tanto, a pesar de haber querido dedicar algo más de tiempo a esta parte, hubiera sido necesario más tiempo y recursos para hacerlo, por lo se considera que el objetivo ha sido cumplido.

Limitaciones

Por último, destacar que debido a la pandemia mundial del COVID-19, lamentablemente no se han podido realizar las últimas pruebas y entrevistas en entornos reales y con los usuarios finales del sistema, algo que hubiera sido una gran experiencia de mucha utilidad, que hubiera permitido hacer nuevas mejoras al prototipo final, y es algo que se espera poder realizar en algún momento.

8.1. Vías de Trabajo Futuro

Tras la realización del proyecto, existen ciertas funcionalidades que se han quedado fuera del alcance del proyecto, pero las cuales pueden ser buenos aportes y mejoras futuras que podrían añadir un gran valor al sistema. Las posibles mejoras que se podrían adaptar al proyecto son las siguientes:

Subida de imágenes: Complementar la utilización de los pictogramas desde la API de ARASAAC con la posibilidad de darle al usuario la capacidad de subir y utilizar sus propias imágenes, lo que facilitaría también el permitir descargar las imágenes seleccionadas desde la propia API. Su implementación pasaría por adecuar el servidor para permitirle alojar y servir imágenes. Estas mejoras darían mas flexibilidad al sistema, permitiendo su uso sin necesidad de que la red WiFi tenga acceso a internet.

Ampliación de la personalización: Añadir nuevas opciones de configuración para las luces LED. Un ejemplo de esto podría ser la posibilidad de dejar al usuario escoger entre toda la gama de colores o los porcentajes de cada una de las fases temporales.

Servidor Externo: Posibilidad de añadir un servidor externo encargado de poder gestionar múltiples dispositivos al mismo tiempo. Esto permitiría controlar la reproducción o la creación de pictolistas desde cualquier dispositivo con acceso a internet, pero aumentaría el presupuesto necesario.

Sonido: El uso conjunto de las luces LED con sonido puede ser también una de las nuevas funcionalidades que aporte nuevo valor al sistema. La colocación de un altavoz permitiría mejorar la percepción del tiempo que dan las luces LED, pudiendo emitir distintos sonidos para indicar cambio de pictogramas o cambio de fases temporales.

Facilitar conexión: Otro de los puntos que se podría mejorar sería el de facilitar la conexión de la Raspberry Pi a la red WiFi, por ejemplo mediante la creación de un punto de acceso propio que permita su configuración remota.

Control Bluetooth: Por último, para añadir más flexibilidad también se podría considerar la posibilidad de controlar el sistema mediante una conexión bluetooth permanente.

Bibliografía

[1] R. Nafkha, "The pert method in estimating project duration," *Information Systems in Management*, vol. 5, no. 4, pp. 542–550, 2016.

[2] C. Lord, E. H. Cook, B. L. Leventhal, and D. G. Amaral, "Autism spectrum disorders," *Neuron*, vol. 28, no. 2, pp. 355–363, 2000.

[3] E. Bleuler, "Autistic thinking.," 1951.

[4] M. J. Allman and I. G. DeLeon, "No time like the present: time perception in autism," *Causes and risks for autism*, pp. 65–76, 2009.

[5] M. Dawe, "Desperately seeking simplicity: how young adults with cognitive disabilities and their families adopt assistive technologies," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 1143–1152, 2006.

[6] J. A. Naslund, K. A. Aschbrenner, and S. J. Bartels, "Wearable devices and smartphones for activity tracking among people with serious mental illness," *Mental health and physical activity*, vol. 10, pp. 10–17, 2016.

[7] T. R. Goldsmith and L. A. LeBlanc, "Use of technology in interventions for children with autism.," *Journal of Early and Intensive Behavior Intervention*, vol. 1, no. 2, p. 166, 2004.

[8] P. M. González, "Sistemas alternativos y aumentativos de comunicación (saac) y accesibilidad: Bases teóricas de los saac," *Puertas a la lectura*, no. 4, pp. 129–136, 2003.

[9] R. W. Schlosser and O. Wendt, "Effects of augmentative and alternative communication intervention on speech production in children with autism: A systematic review," *American journal of speech-language pathology*, 2008.

[10] P. Mirenda, "Toward functional augmentative and alternative communication for students with autism," *Language, speech, and hearing services in schools*, 2003.

[11] J. Ashburner, J. Ziviani, and S. Rodger, "Sensory processing and classroom emotional, behavioral, and educational outcomes in children with autism spectrum disorder," *American journal of occupational therapy*, vol. 62, no. 5, pp. 564–573, 2008.

[12] Velociteam, *PictoTEA*, Accedido el 20 de Enero de 2020.

[13] F. Orange, *DictaPicto*, Accedido el 20 de Enero de 2020.

[14] J. M. M. Laorga, *Agenda de pictogramas*, Accedido el 23 de Enero de 2020.

[15] L. Moreno, *PictogramAgenda*, Accedido el 23 de Enero de 2020.

[16] C. L. Vogel, "Classroom design for living and learning with autism," *Autism Asperger's digest*, vol. 7, no. 1, pp. 30–39, 2008.

[17] M. Cramer, S. H. Hirano, M. Tentori, M. T. Yeganyan, and G. R. Hayes, "Classroom-based assistive technology: collective use of interactive visual schedules by students with autism.," in *CHI*, vol. 10, pp. 1978942–1978944, 2011.

[18] T. Timer, *Time Timer Original 12*, Accedido el 9 de Abril de 2020.

[19] I. Grey, O. Healy, G. Leader, and D. Hayes, “Using a time timer™ to increase appropriate waiting behavior in a child with developmental disabilities,” *Research in Developmental Disabilities*, vol. 30, no. 2, pp. 359–366, 2009.

[20] J. Doe, “Recommended practice for software requirements specifications (ieee),” *IEEE, New York*, 2011.

[21] G. de Aragón, *ARASAAC: Centro Aragonés para la Comunicación Aumentativa y Alternativa*, Accedido el 3 de Febrero de 2020.

[22] G. de Aragón, *API para desarrolladores ARASAAC*, Accedido el 7 de Febrero de 2020.

[23] Alenta, *Alenta.org*, Accedido el 11 de Noviembre de 2019.

[24] EcuRed, *Requisitos no funcionales - EcuRed*, Accedido el 3 de Marzo de 2020.

[25] R. P. Foundation, *Teach, Learn, and Make with Raspberry Pi - Raspberry Pi*, Accedido el 26 de Mayo de 2020.

[26] Arduino, *Arduino - Home*, Accedido el 16 de Diciembre de 2019.

[27] L. Llamas, *Wemos D1 Mini, una genial placa de desarrollo con ESP8266*, Accedido el 16 de Diciembre de 2019.

[28] Adafruit, *Overview | Neopixels on Raspberry Pi | Adafruit Learning System*, Accedido el 13 de Enero de 2020.

[29] A. S. Foundation, *Apache Cordova*, Accedido el 10 de Noviembre de 2019.

[30] O. Foundation, *Node.js*, Accedido el 11 de Noviembre de 2019.

[31] M. Fotache and D. Cogean, “Nosql and sql databases for mobile applications. case study: Mongodb versus postgresql.,” *Informatica Economica*, vol. 17, no. 2, 2013.

[32] Redislabs, *Redis*, Accedido el 22 de Noviembre de 2019.

[33] A. S. Foundation, *Apache Cassandra*, Accedido el 22 de Noviembre de 2019.

[34] I. MongoDB, *La base de datos líder del mercado para aplicaciones modernas | MongoDB*, Accedido el 12 de Noviembre de 2019.

[35] I. Wikimedia Foundation, *Model-view-viewmodal - Wikipedia*, Accedido el 24 de Marzo de 2020.

[36] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. .º’Reilly Media, Inc.", 2011.

[37] O. Collective, *Socket.IO*, Accedido el 3 de Febrero de 2020.

[38] I. Fette and A. Melnikov, “The websocket protocol,” 2011.

[39] R. P. Foundation, *Download Raspberry Pi OS for Raspberry Pi*, Accedido el 16 de Diciembre de 2019.

[40] C. Ltd, *Ubuntu Core | Ubuntu*, Accedido el 10 de Enero de 2020.

[41] M. 2020, *Windows 10 IoT*, Accedido el 10 de Enero de 2020.

[42] F5, *NGINX | High Performance Load Balancer, Web Server, Reverse Proxy*, Accedido el 20 de Febrero de 2020.

[43] C. Rus, *Raspberry Pi 4, características, precio y ficha técnica*, Accedido el 25 de Marzo de 2020.

[44] Waveshare, *4inch HDMI LCD - Waveshare Wiki*, Accedido el 16 de Enero de 2020.

[45] S. Nidhra and J. Dondeti, “Black box and white box testing techniques-a literature review,” *International Journal of Embedded Systems and Applications (IJESA)*, vol. 2, no. 2, pp. 29–50, 2012.

Apéndices

A

Manual de Usuario

A.1. Requisitos mínimos

Los requisitos mínimos para hacer funcionar el sistema son bajos y por lo tanto se puede utilizar en casi cualquier dispositivo. Esto es debido a que se ha priorizado la utilización de hardware de bajo coste, tal y como se ha explicado anteriormente.

Para la implementación del sistema se han utilizado dispositivos con las siguientes prestaciones:

- **Raspberry Pi 4**
- **Windows, Ubuntu, MacOs**
- **Dispositivos Android versión 5+**
- **Dispositivos iOS versión 11+**
- **Navegadores de Ordenadores, menos Internet Explorer 11**

El uso de las Raspberry Pi permite la instalación del software necesario para el funcionamiento del sistema, pero otros modelos o dispositivos de prestaciones similares también pueden ser utilizados para ejecutar el sistema. Por ejemplo el uso de un ordenador con Ubuntu, Windows o macOS sería suficiente para ejecutar el sistema, pero no sería posible la utilización de las luces LED.

Con respecto a los dispositivos móviles u ordenadores, estos son utilizados para entrar a la aplicación de gestión, para la cual no se necesitan equipos con altas prestaciones, y tan solo es necesario que dispongan la posibilidad de obtener acceso a una web y alguna de las versiones de sistema operativo indicadas arriba.

A.2. Manual de instalación

En este capítulo se enseñará como instalar todo el software necesario para el funcionamiento de la aplicación en una Raspberry Pi. Si no se quisiera hacer uso de las luces LED también sería posible instalar el sistema en un equipo con Ubuntu, Windows o macOS, pero esta instalación se centrará en el uso en una Raspberry Pi 4.

1. - Para comenzar será necesario descargar e instalar el sistema operativo Raspberry Pi OS [39], el cual se puede descargar desde la propia web de Raspberry Pi OS, y desde donde también se podrá encontrar una guía de como instalarlo en una tarjeta Micro SD. Una vez instalado el sistema operativo en la Micro SD, se introduce en la Raspberry Pi y se arranca conectándola a la corriente. Si la instalación se ha realizado correctamente comenzará a lanzarse el sistema operativo con normalidad.

2. - Una vez haya acceso al sistema operativo será necesario conectar la Raspberry Pi a la red WiFi en donde la queramos utilizar. Para ello se conectan los periféricos necesarios, teclado y ratón y se introducen los datos de la red WiFi. Una vez está conectada a la red WiFi, es recomendable activar la conexión SSH, para permitir el control remoto de la misma, una conexión FTP para el envío de archivos de forma sencilla, o un programa de control remoto para permitir acceso a la misma desde otro dispositivo.

3. - A continuación se realizará la instalación del servicio de MongoDB para la utilización de la base de datos. En este caso la última versión de MongoDB que es posible instalar en Raspberry Pi es la **2.4.14**, y para ello basta con ejecutar los siguientes comandos en la consola.

```
1.- sudo apt update && sudo apt upgrade
2.- sudo apt install mongodb
3.- sudo systemctl enable mongodb
4.- sudo systemctl start mongodb
```

A.1: Instalación de MongoDB

4. - Para terminar con la instalación de la parte servidor es necesario instalar la versión LTS **12.16.3** de NodeJS. Para la instalación de NodeJS se hará uso de NVM (Node Version Manager) el cual nos permitirá instalar y gestionar las versiones de NodeJS que se utilizarán. También será necesario instalar NPM (Node Package Manager) el cual nos permite instalar los paquetes necesarios para utilizar NodeJS.

```
1.- sudo apt update && sudo apt upgrade
2.- curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
3.- nvm install node
4.- node –version
5.- nvm install 12.16.3
```

A.2: Instalación de NodeJS

Por último, para dejar el servidor listo para ejecutarse, tan solo será necesario entrar en el proyecto que contiene el servidor y ejecutar el siguiente comando para instalar sus dependencias.

```
1.- npm install
2.- npm run start
```

A.3: Instalación paquetes NodeJS

NOTA: Para el control de las luces LED es necesario la instalación de Python 3.8, pero su instalación no será necesaria, ya que viene incluida con la última versión de Raspberry Pi OS.

5. - Por último, para que el dispositivo ofrezca la aplicación de gestión como un servicio accesible desde un navegador, es necesario instalar un servidor web como Apache o NGINX. Para este caso se instalará NGINX como servidor web, y cuya instalación consiste en la ejecución los siguientes comandos.

```
1.- sudo apt update
2.- sudo apt install nginx
3.- sudo /etc/init.d/nginx start
```

A.4: Instalación NGINX

Una vez esté instalado el servidor web, solo faltaría mover el proyecto de la aplicación a la carpeta /var/www/html para que se muestre en los navegadores web.

En el caso de querer ejecutar la aplicación de manera nativa en Android bastaría con enviar el fichero .apk al dispositivo y ejecutarlo directamente para instalarlo. En el caso de iOS es más complejo, ya que no se pueden instalar ficheros directamente, por lo que habría que instalarla directamente desde xCode como developer, como tester, o subir la aplicación a la App Store.

A.3. Manual de uso

En este apartado se realizará una explicación de las funciones principales de la aplicación y como deben de ser utilizadas. Además se explicará todo lo necesario para entender como interacciona la aplicación con el visor de pictogramas y que es necesario para que funcione.

Para el correcto funcionamiento de la aplicación será necesario que la parte servidor del sistema esté ejecutándose, esto se explica más detalladamente en el apartado anterior de Manual de instalación A.2.

A.3.1. Gestión de Listas

Para comenzar a utilizar el sistema será necesario crear una nueva lista que se pueda ser reproducida. Para ello es necesario acceder al menú que se encuentra más centrado y donde se gestiona todo lo relacionado con las listas, es el vista que se muestra en la Figura A.1.

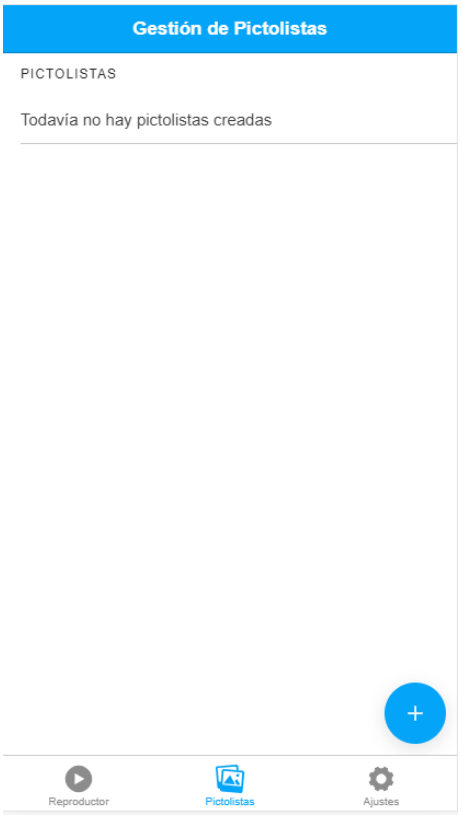

Figura A.1: Menú de gestión de listas

Desde esa misma pantalla se podrán gestionar todas las listas ya creadas, así como crear nuevas nuevas listas. Para crear una nueva lista será necesario pulsar sobre el botón redondeado con el símbolo **+** que se puede encontrar en la parte inferior derecha de la pantalla.

En ese instante aparecerá la vista de creación o edición de listas (Figura A.2), donde se pedirá un nombre y una descripción que la permitan diferenciarse de otras listas ya creadas. Para acabar con la creación de la lista será necesario añadir nuevos pictogramas que reproducir, para eso hay que hacer click sobre el botón de **Añadir Pictogramas**, el cual abre el menú que se puede ver en la Figura 6.2.

Una vez buscados y seleccionados los pictogramas que se desean añadir a la lista, se hace click en el botón **Añadir Pictogramas** que se encuentra en la parte inferior derecha del menú, lo que nos devolverá a la página de creación con la lista de los pictogramas añadidos.

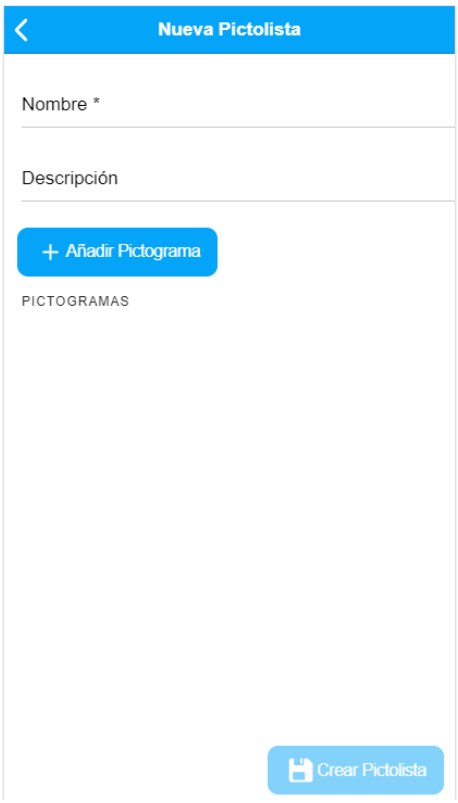

Figura A.2: Menú de creación de listas

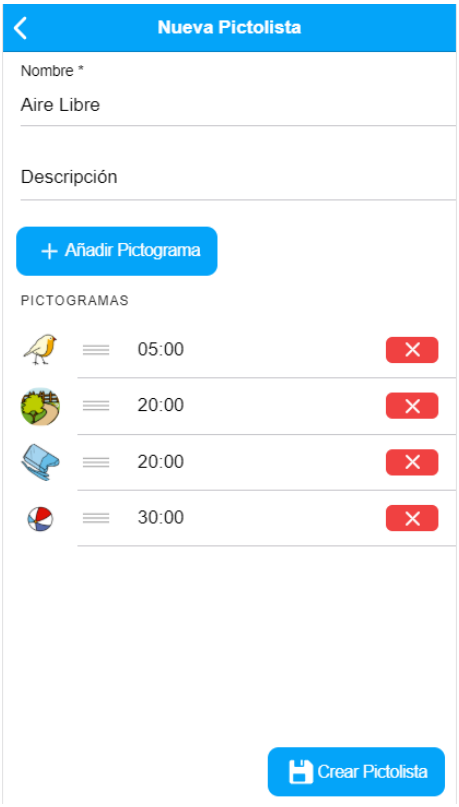

Figura A.3: Ejemplo de creación de lista

En este punto es posible ordenar los pictogramas para variar el orden en el que se van a mostrar en la reproducción, aunque también será posible modificar este orden más adelante. Por último será necesario especificar el tiempo de reproducción de cada pictograma. Una vez rellenados los datos la lista quedaría tal y como se muestra en la Figura A.3.

A.3.2. Reproducción de Listas

Una vez creada una lista llega el momento de reproducirla. Para ello es necesario tener un Visor de Pictogramas activo, el cual es se encuentra en la ruta /**pictogram-display**, es la vista que se lanza automáticamente al encender la Raspberry Pi. En el caso contrario, cuando el visor no esté conectado, no se permitirá reproducir listas y el reproductor principal estará bloqueado.

Una vez haya un Visor de Pictogramas activo se habilitará el botón de reproducir una lista, el cual marcará la lista como activa a la espera de que comencemos la reproducción de la misma. Este botón permitirá que la lista aparezca en el menú principal, el cual se muestra en la Figura A.5, y se permita poder comenzar a controlar la reproducción.

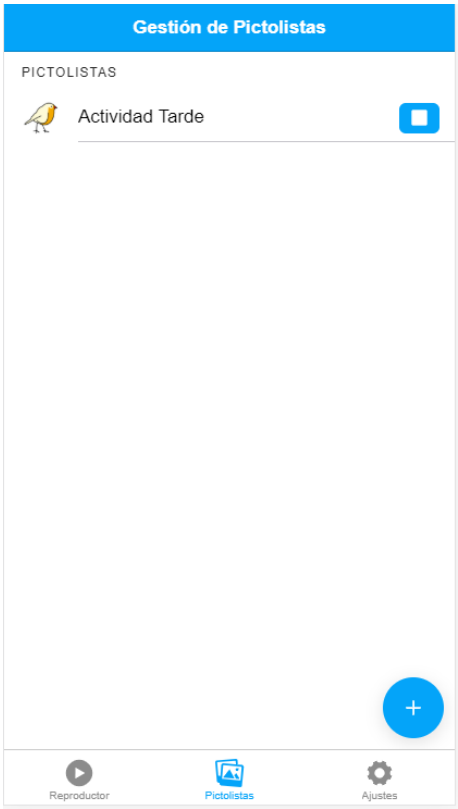

Figura A.4: Reproducir una lista

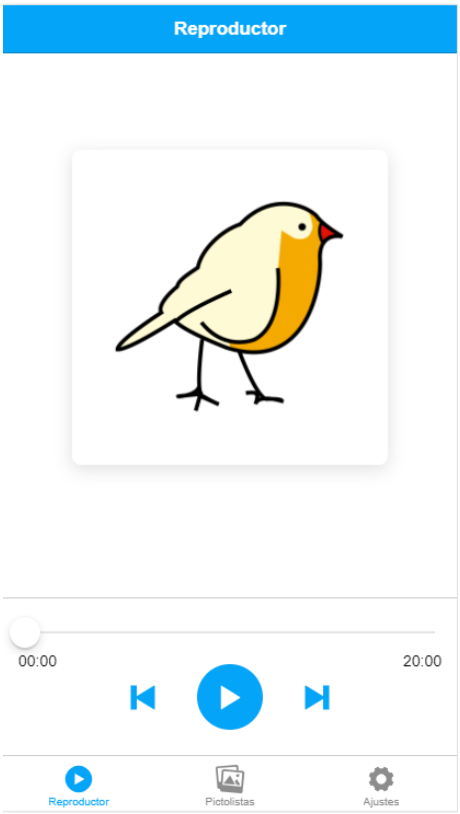

Figura A.5: Menú Reproductor

Una vez se activa una lista, el pictograma que se esté reproduciendo se mostrará en el Visor de Pictogramas (Figura A.6), el cual cambiará y responderá a todas las acciones que realice el usuario desde la aplicación de control.

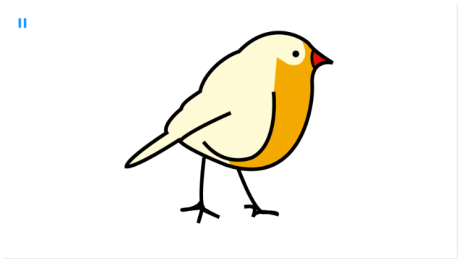

Figura A.6: Visor de Pictogramas

A.3.3. Personalización de las luces

Desde la aplicación es posible configurar y personalizar las luces LED desde el apartado de ajustes ubicado a la derecha de la barra inferior. Desde este menú se podrán configurar algunos ajustes de la aplicación como el idioma, pero la parte más importante es la de personalización de las LEDs.

Existen varias configuraciones posibles que modelan el comportamiento de las luces en la repro-

ducción de listas, como puede ser el patrón, los colores, o el sentido, dando al usuario la posibilidad de escoger la combinación que mejor se adapta al uso que le quiere dar. Para facilitar la personalización al usuario, se muestra en todo momento en la parte superior de configuración de luces una previsualización del comportamiento esperado de las luces.

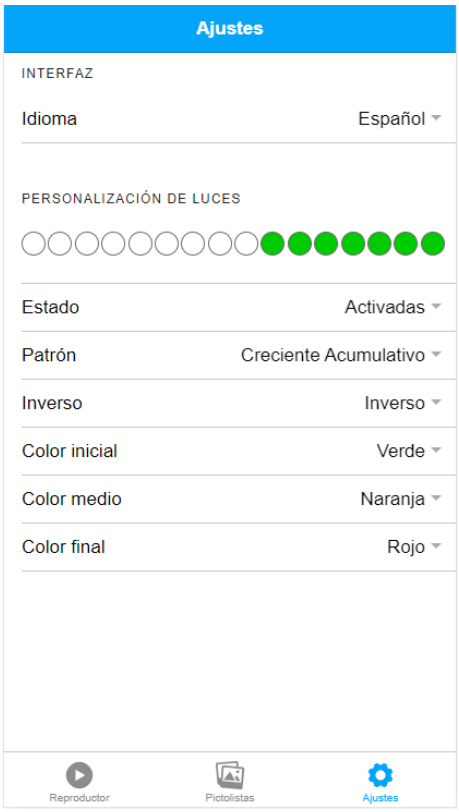

Figura A.7: Menú de configuración

B

Material utilizado

En este capítulo anexo se detallará el inventario de los materiales que han sido utilizados en el proyecto desde la fase de análisis y diseño, por lo que también se especificará si finalmente el dispositivo ha sido utilizado en la solución final, o sin embargo, su utilización fue descartada tras la realización de pruebas de funcionamiento y el diseño final.

Producto	Utilizado	Cantidad	Precio unitario
Raspberry Pi 4 Modelo B - 4GB	Si	1	59.44 €
Fuente de alimentación 15.3W USB-C para Raspberry Pi 4	Si	1	8.97 €
Raspberry Pi Cero W	No	1	10.53 €
Fuente de alimentación Micro USB 5,1V 2,5A	No	1	9.48 €
Adaptador GPIO 2x20 con código de color	No	1	1.51 €
Cable HDMI a micro-HDMI	No	1	2.42 €
Pantalla Táctil 4"HDMI LCD 800x480	Si	1	39.23 €
Pantalla flexible E-Paper HAT	No	1	20.12 €
Tarjeta MicroSD 16Gb Clase 10 U1 SanDisk	Si	2	10.89 €
Tira de LED Ardafruit conectores JST PH	Si	2	12.00 €
Diodo 1N4001	Si	1	0.17 €
Cableado de conexión	Si	35	0.03 €

Tabla B.1: Material utilizado para la realización del proyecto

Cabe destacar que en este inventario de materiales no se ha añadido todo el hardware utilizado en el proyecto, como por ejemplo, dispositivos de trabajo como ordenadores y dispositivos móviles para la realización de pruebas.

C

Descripción de casos de uso

En este anexo se detallan las descripciones de los casos de uso que se han listado en la subsección Funciones del producto 4.2.2, el cual puede ser encontrado en el Capítulo 4 de Especificación y Análisis de Requisitos.

CU-01	**Controlar Reproductor Pictogramas**
Actores	Gestor
Descripción	Control del reproductor de pictogramas
Precondiciones	Está activo un visor de pictogramas y existen listas que reproducir
Flujo Normal	1.- El usuario accede a la vista del reproductor
	2.- El usuario realiza acciones de control sobre la reproducción actual
	3.- El sistema envía estas acciones al servidor y controla la reproducción
Postcondiciones	El sistema responde al control del usuario
Flujo Alternativo	N/A

Tabla C.1: Caso de uso CU-01, Controlar Reproductor de Pictogramas

CU-02	**Reproducir Pictograma**
Actores	Gestor
Descripción	Comienza la reproducción de un pictograma
Precondiciones	Está activo un visor de pictogramas y existen listas que reproducir
Flujo Normal	1.- El usuario accede a la vista de pictolistas
	2.- El usuario selecciona la pictolista que quiere reproducir
	3.- El usuario accede a la vista de control
	4.- El usuario pulsa sobre el botón de reproducir pictograma
	5.- El sistema envía la acción al servidor y comienza la reproducción
Postcondiciones	El pictograma comienza a reproducirse correctamente
Flujo Alternativo	N/A

Tabla C.2: Caso de uso CU-02, Reproducir Pictograma

CU-03	Pausar Pictograma
Actores	Gestor
Descripción	Se pausa la reproducción de un pictograma
Precondiciones	Se está reproduciendo un pictograma
Flujo Normal	1.- El usuario accede a la vista de control
	2.- El usuario pulsa sobre el botón de pausar pictograma
	3.- El sistema envía la acción al servidor y pausa la reproducción
Postcondiciones	El sistema se pausa correctamente
Flujo Alternativo	N/A

Tabla C.3: Caso de uso CU-03, Pausar Pictograma

CU-04	Avanzar Pictograma
Actores	Gestor
Descripción	Se avanza la reproducción hasta el siguiente pictograma
Precondiciones	Se está reproduciendo una lista
Flujo Normal	1.- El usuario accede a la vista de control
	2.- El usuario pulsa sobre el botón de pausar pictograma
	3.- El sistema envía la acción al servidor y avanza la reproducción al siguiente pictograma
Postcondiciones	El sistema avanza el pictograma correctamente
Flujo Alternativo	2a.- Si la lista no tiene más pictogramas se finaliza la reproducción

Tabla C.4: Caso de uso CU-04, Avanzar Pictograma

CU-05	Anterior Pictograma
Actores	Gestor
Descripción	Se reproduce el anterior pictograma en la lista
Precondiciones	Se está reproduciendo una lista
Flujo Normal	1.- El usuario accede a la vista de control
	2.- El usuario pulsa sobre el botón de anterior pictograma
	3.- El sistema envía la acción al servidor y retrasa la reproducción al anterior pictograma
Postcondiciones	El sistema reproduce el anterior pictograma correctamente
Flujo Alternativo	2a.- Si es el primer pictograma de la lista se comienza desde cero su reproducción

Tabla C.5: Caso de uso CU-05, Anterior Pictograma

CU-06	Modificar Tiempo
Actores	Gestor
Descripción	Se modifica el tiempo de reproducción del pictograma
Precondiciones	Se está reproduciendo un pictograma
Flujo Normal	1.- El usuario accede a la vista de control
	2.- El usuario arrastra la barra de tiempo del pictograma
	3.- El sistema envía la acción al servidor y modifica el tiempo de reproducción del pictograma
Postcondiciones	El sistema modifica el tiempo del pictograma correctamente
Flujo Alternativo	N/A

Tabla C.6: Caso de uso CU-06, Modificar Tiempo

CU-07	Gestión de Pictolistas
Actores	Gestor
Descripción	Gestionar las pictolistas del sistema
Precondiciones	Se tiene conexión con el servidor
Flujo Normal	1.- El usuario accede a la vista de listas
	2.- El usuario realiza acciones sobre la gestión de listas
	3.- El sistema envía las acciones realizadas sobre las listas
Postcondiciones	El sistema guarda la nueva lista correctamente
Flujo Alternativo	N/A

Tabla C.7: Caso de uso CU-07, Gestión de Pictolistas

CU-08	Crear Pictolista
Actores	Gestor
Descripción	Crear una nueva pictolista
Precondiciones	Se tiene conexión con el servidor
Flujo Normal	1.- El usuario accede a la vista de listas
	2.- El usuario pulsa sobre el botón crear lista
	3.- El usuario introduce los datos necesarios para crear la lista
	4.- El sistema envía la nueva lista al servidor y la guarda en base de datos
Postcondiciones	El sistema guarda la nueva lista correctamente
Flujo Alternativo	N/A

Tabla C.8: Caso de uso CU-08, Crear Pictolista

CU-09	Editar Pictolista
Actores	Gestor
Descripción	Edición de una pictolista existente
Precondiciones	Se tiene conexión con el servidor
Flujo Normal	1.- El usuario accede a la vista de listas 2.- El usuario pulsa sobre la lista que quiere editar 3.- El usuario introduce los datos que quiere cambiar de la lista 4.- El sistema envía la lista modificada al servidor y la sobre escribe en base de datos
Postcondiciones	El sistema guarda la lista modificada correctamente
Flujo Alternativo	4a.- Si existe una lista con el mismo nombre se pedirá cambiarlo

Tabla C.9: Caso de uso CU-09, Editar Pictolista

CU-10	Eliminar Pictolista
Actores	Gestor
Descripción	Eliminar una pictolista existente
Precondiciones	Se tiene conexión con el servidor
Flujo Normal	1.- El usuario accede a la vista de listas 2.- El usuario pulsa sobre la lista que quiere eliminar 3.- El usuario pulsa sobre el icono de eliminar lista 4.- El sistema envía la lista que se quiere eliminar al servidor y la borra de la base de datos
Postcondiciones	El sistema elimina la lista modificada correctamente
Flujo Alternativo	N/A

Tabla C.10: Caso de uso CU-10, Eliminar Pictolista

CU-11	Reproducir Pictolista
Actores	Gestor
Descripción	Se comienza la reproducción de una pictolista
Precondiciones	Está activo un visor de pictogramas
Flujo Normal	1.- El usuario accede a la vista de listas 2.- El usuario pulsa sobre el icono de reproducir una lista 3.- El sistema comienza la reproducción de la lista marcándola como activa
Postcondiciones	El sistema reproduce la lista correctamente
Flujo Alternativo	N/A

Tabla C.11: Caso de uso CU-11, Reproducir Pictolista

CU-12	Detener Pictolista
Actores	Gestor
Descripción	Se detiene la reproducción de una pictolista
Precondiciones	Se está reproduciendo una pictolista
Flujo Normal	1.- El usuario accede a la vista de listas
	2.- El usuario pulsa sobre el icono de detener una lista
	3.- El sistema detiene la reproducción de la lista marcándola como inactiva
Postcondiciones	El sistema detiene la reproducción de la lista correctamente
Flujo Alternativo	N/A

Tabla C.12: Caso de uso CU-12, Detener Pictolista

CU-13	Obtener PictolistaS
Actores	Gestor
Descripción	Obtención de todas las pictolistas creadas
Precondiciones	Existen pictolistas creadas y hay conexión con el servidor
Flujo Normal	1.- El usuario accede a la vista de listas
	2.- El sistema lista todas las pictolistas creadas
Postcondiciones	Aparece una listas de todas las pictolistas del sistema
Flujo Alternativo	N/A

Tabla C.13: Caso de uso CU-13, Obtener Pictolistas

CU-14	Buscar Pictograma
Actores	Gestor y ARASAAC
Descripción	Se busca un pictograma mediante texto
Precondiciones	El servidor de ARASAAC esta disponible
Flujo Normal	1.- El usuario accede a la vista de creación o edición de pictolista
	2.- El usuario pulsa sobre el icono de añadir un pictograma
	3.- El usuario introduce el texto en el campo búsqueda
	4.- Se realiza una petición a ARASAAC con el pictograma que se busca
	5.- ARASAAC devuelve la lista de pictogramas y se muestran en la aplicación
Postcondiciones	El sistema muestra los pictogramas buscados correctamente
Flujo Alternativo	4a.- Si la API de ARASAAC no está disponible muestra un mensaje de error

Tabla C.14: Caso de uso CU-14, Buscar Pictograma

CU-15	Añadir Pictograma
Actores	Gestor y ARASAAC
Descripción	Se añade un pictograma a una pictolista
Precondiciones	Existe una conexión con el servidor
Flujo Normal	1.- El usuario accede a la vista de añadir pictogramas
	2.- El usuario selecciona los pictogramas que quiere añadir a la lista
	3.- El usuario pulsa el botón para añadir los pictogramas seleccionados
Postcondiciones	El sistema añade los pictogramas a la lista correctamente
Flujo Alternativo	N/A

Tabla C.15: Caso de uso CU-15, Añadir Pictograma

CU-16	Mostrar Pictogramas
Actores	Gestor y ARASAAC
Descripción	Se muestra el pictograma que se está reproduciendo
Precondiciones	Se está reproduciendo un pictograma
Flujo Normal	1.- El usuario accede a la vista de mostrar pictogramas
	2.- Se carga el pictograma que se está reproduciendo desde ARASAAC
Postcondiciones	El sistema muestra los pictogramas correctamente
Flujo Alternativo	N/A

Tabla C.16: Caso de uso CU-16, Mostrar Pictogramas

CU-17	Configurar Luces LED Pictogramas
Actores	Gestor
Descripción	Cambiar la configuración de funcionamiento de las luces LED
Precondiciones	Tener acceso a la aplicación
Flujo Normal	1.- El usuario accede a la vista de configuración LED
	2.- El usuario introduce los datos y la configuración deseada
Postcondiciones	El sistema guarda la configuración introducida correctamente
Flujo Alternativo	N/A

Tabla C.17: Caso de uso CU-17, Configurar Luces LEDs

D

Requisitos Funcionales

En este Capítulo Anexo se detallarán en más profundidad los Requisitos Funcionales listados en la Subsección 4.3.2, el cual puede ser encontrado en el Capítulo 4 de Especificación y Análisis de Requisitos.

Identificador del requerimiento	RF01
Nombre	Estado del sistema
Descripción	La aplicación debe comunicar al usuario en estado actual en el que se encuentra del sistema

Tabla D.1: Requisito funcional RF01

Identificador del requerimiento	RF02
Nombre	Sincronización entre dispositivos
Descripción	El estado del sistema debe estar reflejado de forma sincronizada en todos los dispositivos conectados

Tabla D.2: Requisito funcional RF02

Identificador del requerimiento	RF03
Nombre	Reproducir Lista
Descripción	El sistema debe poder reproducir una lista seleccionada por el usuario

Tabla D.3: Requisito funcional RF03

Identificador del requerimiento	RF04
Nombre	Pausar Lista
Descripción	El sistema debe poder pausar la reproducción una lista

Tabla D.4: Requisito funcional RF04

Identificador del requerimiento	RF05
Nombre	Avanzar Pictograma
Descripción	El sistema debe poder avanzar al siguiente pictograma de la lista

Tabla D.5: Requisito funcional RF05

Identificador del requerimiento	RF06
Nombre	Retroceder Pictograma
Descripción	El sistema debe poder retroceder al anterior pictograma de la lista

Tabla D.6: Requisito funcional RF06

Identificador del requerimiento	RF07
Nombre	Modificar tiempo reproducción
Descripción	El sistema debe de poder modificar el instante de tiempo en el que se encuentra la reproducción

Tabla D.7: Requisito funcional RF07

Identificador del requerimiento	RF08
Nombre	Consultar pictolistas
Descripción	El sistema debe poder consultar las listas creadas

Tabla D.8: Requisito funcional RF08

Identificador del requerimiento	RF09
Nombre	Crear pictolistas
Descripción	El sistema debe poder crear listas de pictogramas

Tabla D.9: Requisito funcional RF09

Identificador del requerimiento	RF10
Nombre	Eliminar pictolistas
Descripción	El sistema debe poder eliminar las pictolistas de ya creadas

Tabla D.10: Requisito funcional RF10

Identificador del requerimiento	RF11
Nombre	Modificar pictolistas
Descripción	El sistema debe poder modificar las pictolistas ya creadas

Tabla D.11: Requisito funcional RF11

Identificador del requerimiento	RF12
Nombre	Buscar pictogramas
Descripción	La aplicación debe permitir hacer una búsqueda de pictogramas por palabras

Tabla D.12: Requisito funcional RF12

Identificador del requerimiento	RF13
Nombre	Conexión a ARASAAC
Descripción	La aplicación debe conectarse a la API de ARASAAC para realizar las búsquedas de pictogramas

Tabla D.13: Requisito funcional RF13

Identificador del requerimiento	RF14
Nombre	Añadir pictogramas
Descripción	La aplicación debe permitir añadir pictogramas a una lista

Tabla D.14: Requisito funcional RF14

Identificador del requerimiento	RF15
Nombre	Borrar pictogramas
Descripción	La aplicación debe permitir eliminar pictogramas de una lista

Tabla D.15: Requisito funcional RF15

Identificador del requerimiento	RF16
Nombre	Modificar pictogramas
Descripción	La aplicación debe permitir modificar los pictogramas de una lista

Tabla D.16: Requisito funcional RF16

Identificador del requerimiento	RF17
Nombre	Asignar tiempo a un pictograma
Descripción	La aplicación debe permitir asignar un tiempo de reproducción a cada pictograma

Tabla D.17: Requisito funcional RF17

Identificador del requerimiento	RF18
Nombre	Seleccionar pictolista
Descripción	La aplicación debe permitir cargar una pictolista para reproducirla

Tabla D.18: Requisito funcional RF18

Identificador del requerimiento	RF19
Nombre	Parar pictolista
Descripción	La aplicación debe permitir parar la reproducción de una pictolista

Tabla D.19: Requisito funcional RF19

Identificador del requerimiento	RF20
Nombre	Personalizar LEDs
Descripción	La aplicación debe permitir personalizar los ajustes de la luces LED

Tabla D.20: Requisito funcional RF20

Identificador del requerimiento	RF21
Nombre	Representación LEDs
Descripción	El sistema debe representar el tiempo restante del pictograma mediante luces LED

Tabla D.21: Requisito funcional RF21

E

Requisitos No Funcionales

En este Capítulo Anexo se detallarán en más profundidad los Requisitos No Funcionales listados en la Subsección 4.3.3, el cual puede ser encontrado en el Capítulo 4 de Especificación y Análisis de Requisitos.

Identificador del requerimiento	RNF01
Nombre	Diseño IU
Descripción	La aplicación debe utilizar una interfaz de usuario sencilla y fácil de entender

Tabla E.1: Requisito no funcional RNF01

Identificador del requerimiento	RNF02
Nombre	Soporte multiplataforma
Descripción	La aplicación debe poder utilizarse desde distintas plataformas y sistemas operativos

Tabla E.2: Requisito no funcional RNF02

Identificador del requerimiento	RNF03
Nombre	Velocidad de respuesta
Descripción	El sistema debe tener un tiempo de respuesta bajo entre la aplicación y el hardware

Tabla E.3: Requisito no funcional RNF03

Identificador del requerimiento	RNF04
Nombre	Soporte multi idioma
Descripción	La aplicación debe poder mostrar los textos en Español e Inglés

Tabla E.4: Requisito no funcional RNF04

Identificador del requerimiento	RNF05
Nombre	Persistencia de la información
Descripción	El sistema debe guardar las listas creadas entre sesiones de uso

Tabla E.5: Requisito no funcional RNF05

Identificador del requerimiento	RNF06
Nombre	Recursos hardware
Descripción	El sistema debe utilzar pocos recursos para poder ser utilizado en hardware de bajo coste

Tabla E.6: Requisito no funcional RNF06

Identificador del requerimiento	RNF07
Nombre	Navegación por la interfaz
Descripción	La navegación por la interfaz debe ser muy sencilla reduciendo el número de pantallas lo máximo posible

Tabla E.7: Requisito no funcional RNF07

Identificador del requerimiento	RNF08
Nombre	Obtención de pictogramas
Descripción	Se tendrá que utilizar un servicio que permita obtener pictogramas de manera sencilla y rápida

Tabla E.8: Requisito no funcional RNF08

Identificador del requerimiento	RNF09
Nombre	Sistema modular
Descripción	El sistema se diseñará para que mejorar sus funcionalidades o añadir nuevas sea un proceso lo más sencillo posible

Tabla E.9: Requisito no funcional RNF09

Identificador del requerimiento	RNF10
Nombre	Personalización
Descripción	El sistema deberá ser altamente personalizable para facilitar la adaptación a diferentes perfiles de usuario

Tabla E.10: Requisito no funcional RNF10

Identificador del requerimiento	RNF11
Nombre	Seguridad
Descripción	El sistema no podrá será accesible por dispositivos externos a su red WiFi

Tabla E.11: Requisito no funcional RNF11

Identificador del requerimiento	RNF12
Nombre	Mostrar errores
Descripción	La aplicación mostrará los errores del sistema de forma clara

Tabla E.12: Requisito no funcional RNF12

Identificador del requerimiento	RNF13
Nombre	Uso de colores
Descripción	Las luces LED utilizarán varios colores para la representación del tiempo

Tabla E.13: Requisito no funcional RNF13

F

Código Control LEDs

En este capítulo anexo se puede observar el código Python de las dos funciones utilizadas para interactuar directamente con las luces LED, una de ellas utilizada para encender LEDs y la otra para apagarlos.

Código F.1: En esta figura se presenta el código correspondiente a las funciones encargadas de encender y apagar las luces LED

```
# Define functions which animate LEDs in various ways.
def colorWipe(start, end, strip, color, wait_ms=3750):
    """Wipe color across display a pixel at a time."""
    for i in range(start, end):
        strip.setPixelColor(i, color)
        strip.show()
        global time_sec
        if(args.currentTime <= time_sec):
            time.sleep(wait_ms/1000.0)
        time_sec = time_sec + wait_ms / 1000

def colorRemove(start, end, strip, color, wait_ms=3750):
    """Wipe color across display a pixel at a time."""
    for i in range(end, start, -1):
        strip.setPixelColor(i, color)
        strip.show()
        global time_sec
        if(args.currentTime <= time_sec):
            time.sleep(wait_ms/1000.0)
        time_sec = time_sec + wait_ms / 1000
```


UAM
UNIVERSIDAD AUTONOMA
DE MADRID